Edgar Miguel Felício Oliveira da Silva

Mestre em Engenharia Electrotécnica e de Computadores

A Multi-Criteria Framework to Assist on the Design of Internet-of-Things Systems

Dissertação para obtenção do Grau de Doutor em

Engenharia Electrotécnica e de Computadores

Orientador: Ricardo Luís Rosa Jardim Gonçalves, Professor Catedrático, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri

Presidente:	Prof. Doutor João Carlos da Palma Goes
Arguentes:	Prof. Doutor Ioan-Stefan Sacala
	Prof. Doutora Teresa Cristina de Freitas Gonçalves
Vogais:	Prof. Doutor Ricardo Luís Rosa Jardim Gonçalves
	Prof. Doutor João Francisco Alves Martins
	Prof. Doutor Manuel Martins Barata
	Prof. Doutor João Pedro Mendonça de Assunção da Silva
	Prof. Doutor Carlos Manuel Melo Agostinho

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

January, 2020

A Multi-Criteria Framework to Assist on the Design of Internet-of-Things Systems

I'll never forget you ...

Acknowledgements

First of all I have to thank all my Family, especially to my uncles Carlos and Maria Jóse Felício, for the unending support, strength in the most difficult times, and for never ever had doubted that I could successfully accomplish this work.

I want to thank my supervisor, Professor Ricardo Jardim Gonçalves, for the opportunity he gave me, to work with him and join his research team. With a constant support, believing in skills and let me freely explore it accordingly with my motivations and instincts.

A special acknowledgement to Manuela Fernandes, for her knowledge and experience to this work, motivation, comments and questions have encouraged, supported and enlightened me to do a better work. Also a special acknowledgement to João, Ricardo, Maria, Paulo and more recently Vanessa for their support and companionship. I wish to acknowledge Fábio Oliveira, Luís Paiva and João Eusébio for their support.

A special thank goes also for the master students to whom I collaborated or had the pleasure of co-supervising, namely João Rodrigues, José Gonçalves, Fernando Rosado, João Ralo and Igor Fernandes.

Then, I would like to thank my colleagues at Group for Research in Interoperability of Systems (GRIS) and Energy Efficiency Research Group for their advices, encouragement and productive discussions that really contributed to this research work.

Finally, my great thanks to Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT NOVA) and to Instituto de Desenvolvimento de Novas Tecnologias (UNINOVA) for hosting this research.

Abstract

The Internet-of-Things (IoT), considered as Internet first real evolution, has become immensely important to society due to revolutionary business models with the potential to radically improve Human life. Manufacturers are engaged in developing embedded systems (IoT Systems) for different purposes to address this new variety of application domains and services. With the capability to agilely respond to a very dynamic market offer of IoT Systems, the design phase of IoT ecosystems can be enhanced. However, select the more suitable IoT System for a certain task is currently based on stakeholder's knowledge, normally from lived experience or intuition, although it does not mean that a proper decision is being made. Furthermore, the lack of methods to formally describe IoT Systems characteristics, capable of being automatically used by methods is also an issue, reinforced by the growth of available information directly connected to Internet spread.

Contributing to improve IoT Ecosystems design phase, this PhD work proposes a framework capable of fully characterise an IoT System and assist stakeholder's on the decision of which is the proper IoT System for a specific task. This enables decision-makers to perform a better reasoning and more aware analysis of diverse and very often contradicting criteria. It is also intended to provide methods to integrate energy consumption-simulation tools and address interoperability with standards, methods or systems within the IoT scope. This is addressed using a model-driven based framework supporting a high openness level to use different software languages and decision methods, but also for interoperability with other systems, tools and methods.

Keywords: Internet-of-Things, Resource-Constrained Systems, Model-Driven Engineering, Interoperability, Multi-Criteria Decision, IoT Systems Assessment.

Resumo

A Internet das Coisas (IdC), considerada a primeira evolução real da Internet, tornou-se imensamente importante para a sociedade devido a modelos de negócios revolucionários com o potencial de melhorar radicalmente a vida humana. Os fabricantes estão focados no desenvolvimento de sistemas embutidos (Sistemas IdC) para diversas finalidades, de modo a responder à nova variedade de aplicações para diferentes domínios e serviços. Com a ágil capacidade para responder a um mercado muito dinâmico na oferta de Sistemas IdC, a fase de desenvolvimento dos ecossistemas da IdC pode ser melhorado. No entanto, a escolha do Sistema IdC mais adequado para uma determinada tarefa actualmente é assenta no conhecimento das partes envolvidas, normalmente baseada na experiência ou na intuição, embora não signifique que a decisão mais adequada esteja a ser realizada. Além disso, a falta de métodos para descrever formalmente as características dos Sistema IdC, capazes de serem usados automaticamente por sistemas, também é um problema, reforçado pelo constante crescimento de informações disponíveis, directamente ligado à propagação pela Internet.

Contribuindo para melhorar a fase de projecto dos ecossistemas da IdC, este trabalho de doutoramento propõe uma estrutura capaz de caracterizar completamente um Sistema IdC e auxiliar as partes interessadas na decisão sobre qual é o Sistema IdC mais adequado para uma tarefa específica. Isto permite que os responsáveis pela decisão executem um melhor raciocínio e uma análise mais consciente dos diversos critérios e muitas vezes contraditórios. Também se destina a fornecer métodos para a integração de ferramentas de simulação do consumo de energia, tal como para estabelecer interoperabilidade com padrões, métodos ou sistemas no âmbito da IdC. Isto será conseguido usando uma *framework* baseada em modelos fornecendo um grande nível de abertura para o uso de diferentes linguagens de programação e métodos de decisão, mas também para a interoperabilidade com outros sistemas, ferramentas e métodos.

Palavras-chave: Internet das Coisas, Sistemas de Recursos Limitados, Engenharia Orientada a Modelos, Interoperabilidade, Decisão Multicritério, Avaliação de Sistemas da Internet das Coisas.

Contents

List of Figures

List of Tables

Glossary

Internet-of-Things Internet-of-Things is a worldwide network of physical objects using the Internet as a communication media. Interconnected objects, services, people, and devices that can communicate, share data, and information to achieve common goals in different areas and applications.

IoT Ecosystem IoT Ecosystem is based on an Internet-of-Things network with sensors and actuators, providing services such as data storage, analysis, using IoT cloud services built upon IoT cloud platforms, within a diverse context of application scenarios (e.g.: Smart Cities, Intelligent Transportation Systems, Domotics (Smart Buildings), Industry 4.0, among others).

IoT System IoT System defines a "thing", a device executing a task for the Internet-of-Things (IoT), as a whole. It is considered that an IoT System is composed by two core parts and a third one if available. The two core parts are the hardware components and the software (Operating System (OS) and application). The third part is an energy profile, built upon, from the two mandatory IoT System parts. An IoT System is by its nature a Resource-Constrained System (RCS).

Resource-Constrained Systems Resource-Constrained System is a device that has small size, low-power operations, limited processing and storage capabilities, and that often runs on batteries. They also present other particular features such sensing the environment and wireless communication.

System Accordingly to online dictionaries and author point of view, a system can be defined as a "group of items forming a unified whole"[1], "a set of connected things or devices that operate together", or "a set of computer equipment and programs used together for a particular purpose"[2].

Acronyms

ADC Analog-to-Digital Converter.

AHP Analytic Hierarchy Process.

AMQP Advanced Message Queuing Protocol.

API Application Programming Interface.

ATL Atlas Transformation Language.

AWS Amazon Web Services.

CASE Computer-Aided Software Engineering.

CIM Computation Independent Model.

CoAP Constrained Application Protocol.

CPS Cyber-Physical System.

DDS Data Distribution Service.

DiY Do it yourself.

ELECTRE Elimination and Choice Expressing the Reality.

EMF Eclipse Modeling Framework.

ETSI European Telecommunications Standards Institute.

GPS Global Positioning System.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

ICT Information and Communication Technology.

IEC International Electrotechnical Commission.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

IoT Internet-of-Things.

IoTSAC IoT Systems Assessment Core.

IoTSAG IoT Systems Analysis Generic.

ISA International Society of Automation.

ISM Industrial, Scientific, and Medical.

ISO International Organization for Standardization.

IT Information Technology.

ITU International Telecommunication Union.

ITU-T ITU Telecommunication Standardization Sector.

JMS Java Message Service.

LoRa Long-Range.

LTE Long-Term Evolution.

M2M Machine-to-Machine.

MCAM Multi-Criteria Analysis Meta-Model.

MCDM Multi-Criteria Decision-Making.

MDA Model-Driven Architecture.

MDD Model-Driven Development.

MDE Model-Driven Engineering.

MEMS Micro-Electro Mechanical System.

MOF Meta-Object Facility.

MOM Message-Oriented Middleware.

MQTT Message Queuing Telemetry Transport.

NB-IoT Narrowband IoT.

nesC Network Embedded Systems C.

NFC Near-field communication.

OASIS Organization for the Advancement of Structured Information Standards.

OGC Open Geospatial Consortium.

OMG Object Management Group.

OS Operating System.

OSS Open-Source Software.

PIM Platform Independent Model.

POC Proof-of-Concept.

PROMETHEE Preference Ranking Organization Method for Enrichment Evaluation.

PSM Platform Specific Model.

PtoP Point-to-Point.

QoS Quality-of-Service.

QVT Query/Views/Transformations.

RAM Random-Access Memory.

RCS Resource-Constrained System.

RCSH Resource-Constrained System Hardware.

RCSM Resource-Constrained System Meta-Model.

REST Representational State Transfer.

RFID Radio-Frequency identification.

SensorML Sensor Model Language.

SMTP Simple Mail Transfer Protocol.

SPI Serial Peripheral Interface.

SRAM Static Random-Access Memory.

SSN Semantic Sensor Network.

STOMP Streaming Text Oriented Messaging Protocol.

SUS System Under Study.

SysML Systems Modeling Language.

TCP Transmission Control Protocol.

TCP/IP Transmission Control Protocol / Internet Protocol.

UART Universal Asynchronous Receiver/Transmitter.

UDP User Datagram Protocol.

UML Unified Modelling Language.

USB Universal Serial Bus.

W3C World Wide Web Consortium.

WLAN Wireless Local Area Network.

WMAN Wireless Metropolitan Area Network.

WPAN Wireless Personal Area Network.

WSN Wireless Sensor Network.

WWAN Wireless Wide Area Network.

XMI XML Metadata Interchange.

XML Extensible Markup Language.

XMPP Extensible Messaging and Presence Protocol.

Chapter 1

Introduction

A System can be defined as a "group of items forming a unified whole" [1] "set of connected things or devices that operate together", or "a set of computer equipment and programs used together for a particular purpose" [2]. Consequently, an IoT deployment is an IoT System, considering that each device interacting in a deployment is a thing/item, and this is the common meaning used in the IoT scope literature. Taking into consideration a single device, it is by its own a system, composed by different hardware components interacting among them and running one or more programs/firmware.

Hereupon, IoT complete system is referred as IoT Ecosystem, IoT Deployment or System of IoT Systems, and an IoT device as IoT System or RCS.

The Internet-of-Things (IoT) today is a reality. It is a highly heterogeneous environment [3], composed by a vast number of "things" (devices, sensors, smart objects, etc.) and sometimes called "Internet of Objects", which refers to uniquely identifiable objects and their representation in an Internet-like structure. Some studies, predict that there will be around 30 Billion devices wireless connected to the IoT in the next years [4, 5]. The emerging IoT-related technologies have been creating an idea of a global, dynamic network infrastructure where physical and virtual "things" communicate and share information among each other. Powered by advances in microelectronic technologies, that led to smaller and cheaper devices, capable of performing major tasks than before.

However, IoT is still in an early stage and many challenging issues still need to be addressed, both at technological, as well as, at social level. With the potential to slowdown IoT development, they need to be solved before IoT paradigm becomes widely accepted. For instance, from a technological point of view, questions have been raised on how to fully establish devices interoperability (adaptation and autonomous behaviour), as well as, new problems concerning network aspects (unique IP addresses, scalability, etc.). Regarding social aspects it is important to guaranty trust, privacy and security to the user.

It should be assured that personal information is used accordingly and that there is no unauthorized disclosure of information [6–9].

1.1 Motivation

The Internet-of-Things (IoT) technology is becoming more and more important to society, with diverse and novel business models giving to people a more truly interaction with the surrounding world. IoT has applicability in many areas, such as Smart Cities, Intelligent Transportation Systems, Domotics (Smart Buildings), Industry 4.0, among others [3].

These "smart systems", i.e.: IoT deployments, IoT Ecosystems, are normally built upon devices with a very specific characteristic — Resource-Constrained. These "things", "objects", are devices characterised with high resources constraints mainly due to low power, small processing and storage capabilities.

Manufacturers are engaged in developing new embedded systems for different purposes, addressing the variety of application domains and services. This diversity unlocks the use of a wide set of hardware platforms and software solutions (OS as implementation techniques). IoT Systems perform different tasks, such as, sensing, radio messages, mathematical computation, etc. each one with a distinct influence on energy consumption. Changing the device/hardware even more uncertainties arises. Not only will the energy consumption, but also with execution time as well as development time. Cost, size and weight factors will also be different when an implementation platform is change.

Consequently some scenarios can then be established: which are the more suitable devices (hardware platforms) to accomplish a certain task, or is it possible/feasible to build a new hardware through the integration of distinct components available on other devices, or which is the best way to programme the execution of a task. Researchers, domain practitioners, engineers are developing OSs to get more performance for less energy consumption. Computation requirements, flexibility, fault tolerance, high sensing fidelity, low-cost and rapid deployment are other characteristics expected when thinking in deploying IoT Systems [10–12]. Combining or fulfil several requirements is not an easy task. Applications differ from each other making the selection of a perfect solution hard to come by.

Therefore, engineers are facing a difficult task, i.e.: to choose a solution between all available ones that best fulfil their intentions. Select a solution normally involves the analyses of criteria. In this case, such criteria could be performance, storage, energy consumption, cost, size, weight, available resources, among others. The importance of a requisite/feature change from task to task. Make a correct, accurate decision depends many times on multiple criteria, which is a tough challenge for human beings [13]. Besides, the lack of formal descriptions to describe IoT Systems characteristics/features, capable of being used by applications in an automatic way, is also an issue. Some manufacturers provide in their websites ways to select a product from a set of choices. Products

are displayed with some detailed (number of features), but limited to hardware characteristics. Interaction is completely handmade, with no way to apply user's preferences between criteria. Information data, when possible, can be collected only by user's action and change depending on the performed query. Two examples can be found in [14, 15].

1.2 Vision and Research Approach

The research work envisions the proposal of a framework capable of fully characterise an IoT System and assist stakeholder's on the decision of which is the proper IoT System for a specific task. The framework also intends to provide methods for energy consumption-simulation tools integration and address interoperability with standards, methods or systems within the IoT scope.

Literature has been more focus on study functional, behaviour aspects and on activities, interactions within an IoT Deployment [16–18], rather than on IoT Systems itself. Stakeholders have been selecting an IoT System to perform a certain task based on their knowledge, normally from lived experience or intuition, although, it does not mean that a proper decision is being made. This task, selection, identification of the more suitable IoT System is being somehow neglected during the design phase of an IoT Ecosystem.

In a highly diversified market-offer (manufacturers are providing numerous device solutions), engineers could use a multi-criteria framework to simulate and analyse hardware and software solutions for small parts of a global system. It will allow IoT-related technologies and solutions to be proven, tested before deployment into real world. That enables engineers to perform a more suitable design of their IoT Ecosystems.

Figure 1.1 depicts the research work approach, displaying literature themes used as groundwork and in which way these help to achieve the expected outcomes. To better understand and respond to the identified research questions (presented in Section 1.4), the work research was split into three objectives: Formally describe an IoT System; characterise/analyse possible IoT System solutions; and properly make a decision regarding which is the more suitable solution(s).

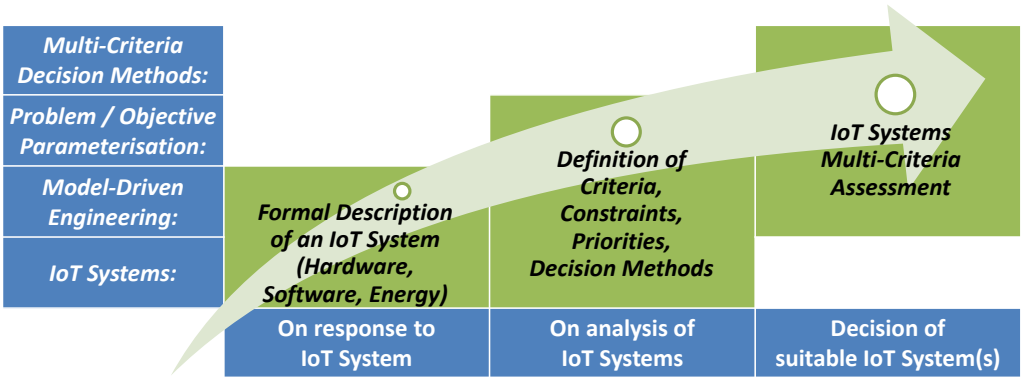


Figure 1.1: Research Approach: The Path to Select a More Suitable IoT System.

First, "**on response to IoT System**", the heterogeneity nature of IoT deployments is addressed. Markets are offering a wide device diversity, with a very specific characteristic — Resource-Constrained. Methods to formally describe these imperative pieces (IoT Systems) are needed, independent of hardware platforms, as well as independent from programming languages — Platform Independent. In this way, Resource-Constrained Systems (RCS) should be study carefully and Model-Driven Engineering (MDE) techniques study and applied, since model driven approaches revealed to be a common ground in literature [19–23].

Second, "**on analysis of IoT Systems**", addresses the characterisation of the problem that needs to be solved. As a result from the previous point, IoT Systems features are formally defined, allowing a smooth integration with other systems/tools. The problem parameterisation consists on defining the objective, criteria, constraints, etc. Once more, MDE gains here an important role. MDE tackles systems complexity through simplification and formalisation techniques during system life cycle (i.e. from design to deployment, passing by construction, operation, modification, etc.) [23], and facilitates the interoperability among tools.

Third point addresses the "**decision of suitable IoT System(s)**". Thematic studied focus on Multi-Criteria Decision-Making (MCDM), one of the most widely used decision methodologies in sciences, business, governmental and engineering worlds. MCDM methods improve decisions quality, by making the decision-making process more explicit, rational, and efficient [24, 25].

1.3 Adopted Research Method

Merriam-Webster dictionary [26] defines the scientific method as "*the principles and procedures for the systematic pursuit of knowledge involving the recognition and formulation of a problem, the collection of data through observation and experiment, and the formulation and testing of hypotheses*". Scientists use the scientific method as a logical scheme to search answers to questions posed within science. Being able to yield scientific theories, including both scientific meta-theories (theories about theories), as well as, the theories used to design tools for producing theories (instruments, algorithms, etc.) [27].

There are more than one recognise scientific method. There is no universal, no formal "scientific method" to follow, each one has its own variants. The process of investigation is often referred in many textbooks and science courses, as a linear set of steps through which a scientist moves from observation through experimentation to a conclusion. It is important that researchers keep in mind that a new piece of information (i.e. idea, test results, etc.), might cause a scientist to rethink the investigation process and repeat steps at any point. Also, it is not always required to start with a question, and sometimes it is not needed to have experiments. Instead, the scientific method is a more dynamic and robust process [28, 29].

The choice of which research method (instantiation of the scientific method) to use is personal and depends on the scientist and the nature of the question addressed. It is also important to choose a research method which is within the limits of time, money, feasibility, ethics and the availability of the chosen scientific measurements to get a correct conclusion [30]. To address this PhD work, the selected research method is based on a 9 step method that, as suggested in [27], considers being important the continuous re-examination and self-correction, by testing the hypothesis that has is origins in the researcher's background knowledge. Next, the adopted method, presented in Figure 1.2, is described in more detail:

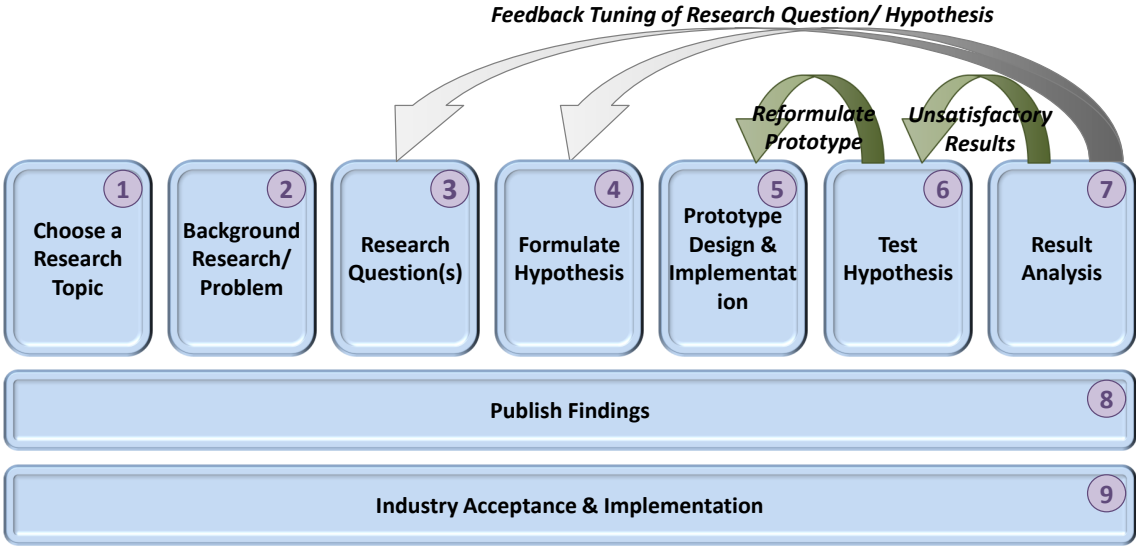


Figure 1.2: Adopted Research Method.

1. **"Choose a Research Topic"**: Since the student has his own research area, i.e.: his concern area, the fundamental is to set the topic of interest where the research will focus. In fact this can be seen as a preliminary step towards the real method.

2. **"Background Research/Problem"**: A background research provides means to properly formalise the problem, as well as, gather solid bases regarding previous and similar work. This can be reach through literature review, informal discussions in conferences and/or workshops, etc. Also, any study will allow assemble of information concerning similar approaches, and so, point out the differences to what will be done [31].

3. **"Research Question(s)"**: This is one of the most important steps, if not the most important one. This step describes what the researcher is interested in, what he/she have/wants to know, by posing the question(s) in the context of the existing knowledge. Also, it will scope the entire work, and will never be revisited in the same research loop until a conclusion can be obtained from the results analysis [31]. The Research Question can be split in secondary questions, to tighten the focus of the

study, but always keeping in mind that all questions have to be about something that can be measure. Consequently, to be confirmed or denied, i.e.: answered. Section 1.4.1 addresses this step.

4. **"Formulate Hypothesis"**: A scientific hypothesis is an educated guess about how things work: "If____*[I do this]* ____, then ____*[this]* ____ will happen.". The hypothesis must be stated in a declarative format, in a way that can be easily measure, and of course, the hypothesis should be constructed in such way that helps answering the original question. Additionally, the hypothesis should be simple, specific and conceptually clear, since ambiguity would make verification almost impossible. It also should be capable of verification through the use of methods and techniques for data collection and analysis [31]. Other aspect, as illustrated in Figure 1.2, is that the hypothesis can be revisited and reformulated in case of unsatisfactory results during further stages of the scientific method. Section 1.4.2 presents the hypothesis of this work.

5. **"Prototype Design & Implementation"**: Once the hypothesis is defined, it is required to design a prototype to prove whether it is true or false. This includes planning in detail all steps of the experimental stage, and the design of the system architecture (the prototype itself), often related to engineering research. A complete validation of the hypothesis is achieved by running more than once the same experiment in order to prove that results are consistent. Therefore, this step relates directly to the design of an experiment in a controlled environment. However, at this stage is important the prototype design, but also the thesis validation method. Namely, scenarios and/or use cases definition (presented in Chapter 8). This can lead to definition of intermediate milestones, not forgetting that the plan must be feasible.

6. **"Test Hypothesis"**: This step is where the researcher actually puts to test the experiment, to either prove or disprove the hypothesis. It includes the implementation of a prototype, data gathering and execution of tests according to the pre-established validation method. Remarks concerning implementation must be induce, since the research may find evidence that the prototype needs rectifications, forcing the jump to the previous step.

7. **"Results Analysis"**: Once the testing phase is finished, is time to evaluate by means of qualitative and quantitative data analysis the factual results. It is important to have a critical spirit and promote discussions regarding literature, research objectives and research questions [31]. At this moment conclusions should be drawn, in order to be decided the next step. In case, hypothesis fails, it must be rejected and either abandoned or modified by jumping to step 4. Whether, it just needs to be re-tested, the next step should be the previous one.

8. **"Publish Findings"**: To be a contribution, the research results must be published. Results are published and shared with peers from the scientific community, allowing verification of findings and enabling others to continue research into other areas. Despite of appearing as one of the final steps, publications should cover not only the final findings but also the intermediated ones. Section 9.2.2 reports the accomplished publications.

9. **"Industry Acceptance & Implementation"**: Besides sharing the work developed with peers from the scientific community through accomplished publications, each research work should also be validated by Industry. Developed methodologies and prototypes should be transferred, accepted, and at the end applied to Industry. In this sense, the proposed work application in Industry will be presented in Section 8.3.2.

1.4 Research Problem and Hypothesis

1.4.1 Research Question

Every day that passes we find ourselves more and more living surrounded by systems sensing the environment. Our daily life (e.g.: at our homes, neighbourhood, workplaces, restaurants, transportation utilities) are increasingly being fulfil with sensing devices capable of providing additional information about who/what surrounds us or even what we can expect in the next corner.

Advances in microelectronic technologies have brought small and cheap devices. Manufacturers want to address different application domains and services to target different IoT deployments, in these sense they are developing devices for different purposes. Although, these devices are resource constrained, for example in terms of low power, small processing and storage capabilities [32].

The diversity of possible solutions for a single application/service, brings up the lack of a methodology to analyse different solutions, taking into account distinct criteria, as well as, the capability to propose a more suitable solution regarding a set of prerequisites.

Considering the chosen topic, the main research question that gives motto for this research work is:

> *"How can Internet-of-Things Systems be designed to optimise the matching with the operating environment?"*

To assure the research focus and targeted results, the major question is split according to three major incognita:

- On response to IoT System

  - **Q1.1:** "Which methods could be applied or develop to formally describe an IoT System (hardware, software, energy)?"

- On analysis of IoT System solutions

  - **Q1.2:** "Which methods could be applied or develop to assist in IoT System assessment?"

- Decision of suitable solution

  - **Q1.3:** "Which multi-criteria decision framework would provide a suitable decision support for the design of IoT Systems?"

1.4.2 Hypothesis

The previous subsection pointed out a question which comes from the identified problem: in a world with a diversity of possible solutions, engineers are facing the difficulty to choose in a more conscious way, a more suitable solution on how to implement and/or improve a certain task.

The main reason why it is important to have a multi-criteria framework, capable of simulate and analyse different hardware and software solutions is mainly to assist developers, in a way that they can improve their own or verify other available solutions, by examining the performance according to a set of features. Engineers have in their hands many different devices, each one built with a very specific purpose. This diversity is basically due to different hardware components (e.g.: microcontrollers, radio, sensing) and programming. Confronted with extremely heterogeneous environments, created from numerous distinct devices, engineers should use modelling techniques to assist in hardware and software formalisation, so it can be then used by tools/methods for simulation and analysis of possible solutions.

The adopted hypothesis for this work is:

> *"If in an Internet-of-Things system design, engineers could use a multi-criteria framework to simulate and analyse hardware and software solutions, then proper solutions could be selected and applied, leading to a more suitable design of an Internet-of-Things System."*

Figure 1.3 displays the presented research question and its derived sub-research questions, to allow a better understand and solidify the main research question. Question which leads to the proposed hypothesis, in response to the introduced problem.

1.5 Thesis Plan Outline

This dissertation document is organised in five core blocks arranged along nine sections, as depicted in Figure 1.4, including also sections for references and appendixes. The

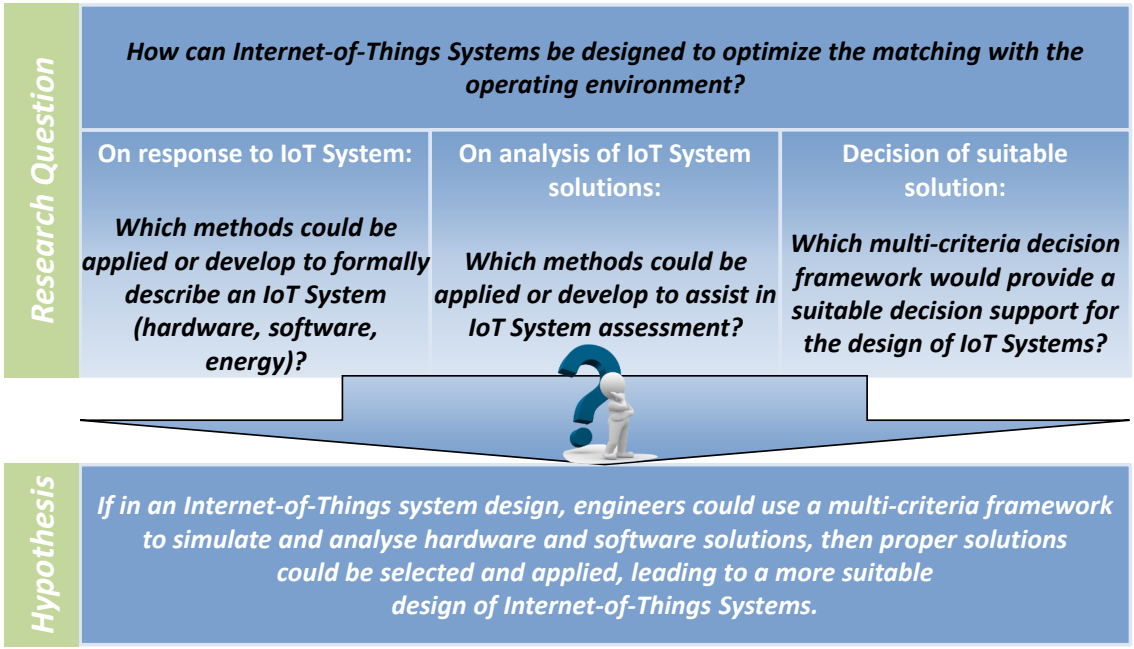


Figure 1.3: Research Question and Hypothesis.

first core block, **Core Block I: Introduction**, gives an initial elucidation regarding the nature of the work presented in this document. The second core block — **Core Block II: Background**, focus on background studies regarding this research work scope, but also on background thematics that gives support to the work developed. The third core block, **Core Block III: Contributions**, presents the work contributions. Then, **Core Block IV: Validation**, describes application scenarios and respective outcomes and exploitation of the results. Finally, **Core Block V: Conclusions** presents final considerations and point out future contributions that could improve the work developed.

1.5.1 Core Block I: Introduction

This core block is Chapter 1 — "**Introduction**", i.e. the current chapter. Begins with a contextualisation to Internet-of-Things (IoT) theme, highlighting some identified problems as part of the author's motivation and consequent vision and research approach for this work' contributions. It is also presented the research method used by the author, followed by the main research question that had risen from the identified problem, from which derives three sub-research topics to allow a more assertive response to the overall problem. A hypothesis is proposed to address the main research question.

1.5.2 Core Block II: Background

The core block, ***Background***, composed by three chapters serves as basis for this research work, and in this sense, is of major importance. Chapter 2, **Internet-of-Things (IoT)**, will address IoT challenges, barriers and trends focus on IoT resource-constrained nature,

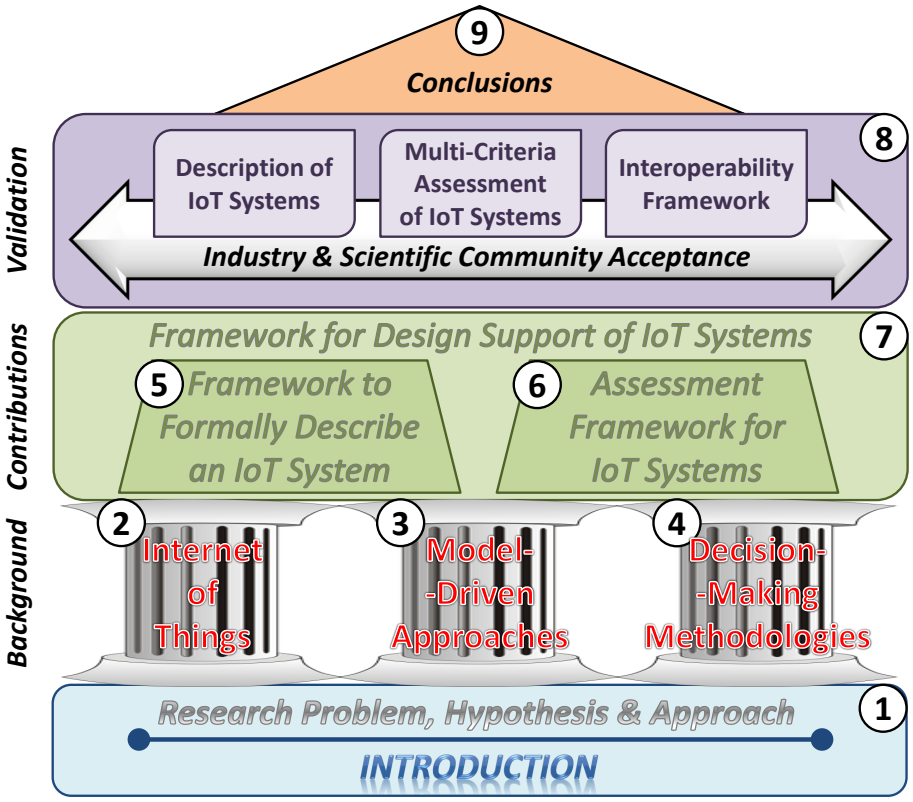


Figure 1.4: Thesis Plan Outline

devices diversity and standards. Followed by Chapter 3, **Model-Driven Approaches**, a solid-ground concept and has revealed to be a common ground in literature [19, 21–23]. Chapter 4, **Decision-Making Methodologies**, focus on MCDM, a thematic considered by governmental, business, engineering and sciences worlds as one of the most important decision methodologies.

1.5.3 Core Block III: Contributions

This core block, ***Contributions***, is also composed by three Chapters which are described next:

- **Framework to Formally Describe an IoT System:** is the first chapter (Chapter 5) of this core block. It starts by presenting a literature review on "what" and "how" descriptions, specifications are made regarding IoT deployments thematics, focus mainly on devices hardware representations. Then, the proposed framework to formally describe an IoT System is presented. This framework uses a model-driven approach to formally specify IoT Systems;

- **Assessment Framework for IoT Systems:** is Chapter 6, presenting the framework to perform IoT Systems assessment. First, a literature review is presented showing how MCDM methodologies have being applied to IoT independently of the area

(e.g.: security, communications, platforms, etc.), since the proposed framework is a novel-framework focused on an existing problem not yet addressed — IoT Systems multi-criteria assessment. Specification models are presented, as well as, the proposed assessment methodology for a multi-criteria analysis of IoT Systems. The framework enables the use of different MCDM methods, even new or user-defined within the proposed assessment methodology;

- **Framework for Design Support of IoT Systems:** is the final chapter (Chapter 7) of this core block. It presents the conceptual approach for design support of IoT Systems. To materialise the conceptual approach is proposed a framework that embraces, aggregates the two previous contributions providing a mechanisms to assist stakeholders on the decision of IoT Systems, enabling conscious and justifiable decisions upon more suitable IoT System(s), strengthened with modules that enable integration, interoperability with systems and tools (e.g. Standards, energy-assessment tools).

1.5.4 Core Block IV: Validation

This core block, ***Validation***, correspond to Chapter 8 "*Implementation and Hypothesis Validation*". In this Chapter, scenarios were described that allow to demonstrate the proposed contributions functionality and to test the design idea of the conceptual approach for design support of IoT Systems. It follows technical implementations addressing each test-case, with IoT Systems formal representation, multi-criteria assessment and the identification of the more suitable IoT System during a Smart Building Ecosystem design. It is also presented a test-case to validate the interoperable nature of the proposed framework, but also evidence that this work contributes with a very useful tool to assist in the design of IoT Systems with the capability to be integrated with IoT Ecosystems management systems. Providing a set of structure data (IoT System information) needed to define observations, behaviours, functional aspects and processes. It is also addressed the acceptance of the work presented, both by peer researchers and also industrials. During this research work the author has published in a number of international conferences and scientific journals, and adds his work put-to-test in an industrial scenario.

1.5.5 Core Block V: Conclusions

This core block is Chapter 9 — "**Conclusions**", in which the main conclusions of the developed work are presented, enhancing the novelty of this research. Then, it is point out future contributions that could improve the work developed and to promote further developments.

This is followed by an extensive list of bibliography and some appendixes.

Chapter 2

Internet-of-Things (IoT)

The Internet-of-Things (IoT) is today a reality. IoT has become immensely important because it is the first real evolution of the Internet, leading to revolutionary applications with the potential to radically improve humankind. It has brought to Internet sensing capabilities allowing us to become more proactive and less reactive. IoT will change everything — including ourselves [7].

IoT deployments are flourishing everyday all over the world and in many application areas. The constant creation of new technologies and innovation of existing ones, has been leading us to a new world — a "Smart World". This is based on systems such as Smart Cities, Intelligent Transportation Systems, Domotics (Smart Buildings), Industry 4.0, etc., often call Smart Systems. Systems that consist in the integration of computation units, networking and physical systems [33], built over Wireless Sensor Network (WSN) which are formed by actuators, gateways and distinct autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc.

IoT is considered to be a dynamic global network infrastructure where physical and virtual "things" communicate and share information among each other (see Figure 2.1). A highly heterogeneous environment, composed by a vast number of "things" (devices, sensors, smart objects, etc.), that are conceived by an even more number of manufacturers, and designed for much different purposes [6].

Sometimes it is also called the Internet of Objects, referring to a uniquely identifiable objects and their representation in an Internet-like structure. Some studies have predicted the amount of devices connected to the IoT by 2020. Gartner, Inc. states than there will be almost 26 billion devices [4] and ABI Research defends that will be more than 30 billion devices wireless connected to the IoT [5].

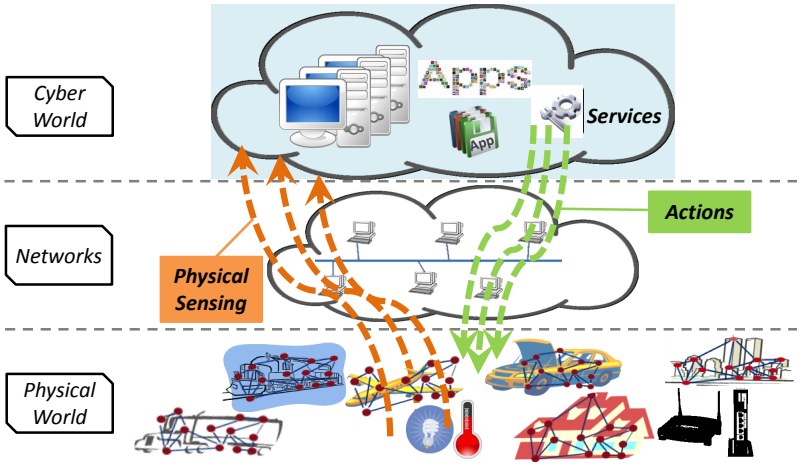


Figure 2.1: Overview of an IoT Ecosystem.

2.1 Challenges, Barriers and Trends

As stated, it is predicted that the number of digitally identifiable, potentially linked, electronic devices will increase rapidly. This encourages new commercial IoT deployments, disruptive opportunities and new services to emerge. Consequently the amount of data created by the IoT will grow exponentially and will continuously rising into IoT Big Data [6].

The existence of several "things" providing different types of information (temperature, humidity, light, noise, Location GPS, pressure, etc.), requires new ways to capture, storage, analyse, visualise, etc. It also brings large and heterogeneous amount of structured or unstructured data.

The IoT is also based on a dynamic network infrastructure, with a highly heterogeneous environment, multi-manufactures and multi-service [3]. These aspects bring together good ingredients for interoperability problems. A high level of interoperability needs to be reached at the communication level, as well as, at service and information levels. Crossing different platforms, but established on a common ground [34].

In [8], the identified open issues in the IoT are standardisation, communication, scalability, security and privacy. In all mentioned topics a remark is made — the energy consumption. Devices have low energy capabilities to perform exhaustive computations. The white paper [7] refers as challenges to IoT development, the deployment of Internet Protocol version 6 (IPv6), energy consumption and standards especially for security, privacy, architectures and communications. The research work presented in [33] distinguishes some needs for future IoT Ecosystems. These are scalability, energy consumption, standards, architectures, big data, privacy, security, trust and real time-based solutions. In [9] the authors point out some key features that IoT must support in the future. These features are: heterogeneity; scalability; data exchange; energy consumption; tracking and localisation; self-organisation; semantic interoperability and data management; security

and privacy. It is also discussed what are the main limiting factors to the IoT development. These factors were identified as being the heterogeneous devices nature, the energy efficiency and devices dimension.

Through this small review, it is then possible to infer that mainly the IoT development issues are: scalability; standardisation, specifically to security and privacy; and energy consumption.

One of the main drivers of the IoT investigation is the European Commission, through its new research programme call Horizon 2020 (H2020) — The EU Framework Programme for Research and Innovation (R&I). In which the Information and Communication Technology (ICT) Cross-Cutting Activities embraced in 2015 the topic ICT 30 — 2015 Internet of Things and Platforms for Connected Smart Objects. It was intended that this topic cut across several LEIT-ICT challenges (smart systems integration, Cyber-Physical System (CPS), smart networks, big data) and brings together different generic ICT technologies, such as, wireless networks, low-power computing, adaptive and cognitive systems [35]. For 2016 and 2017 the Cross-cutting activities had a special call for the Internet-of-Things with focus for Large Scale Pilots, IoT Horizontal activities, and R&I action on IoT integration and platforms [36]. From 2018 to 2020, under ICT scope was proposed a specific call for IoT (ICT-27-2018-2020). Although, IoT concept is referred in other calls, involving big data and large-scale test-beds (ICT-11-2018-2019), digital innovation and interoperability for industry and services (DT-ICT-10-2018-19), and cyber-security (SU-ICT-02-2020). It was also proposed joint calls with other world countries. A call with Japan focused on advanced technologies combining Security, Cloud, IoT, Big Data for a hyper-connected society. With Korea focus on Cloud, IoT and Artificial Intelligence technologies [37].

IoT is a relevant part in the new generation of information technology, covering an extremely wide scope. IoT potential is great, but it is also an important challenge to society due to the vast range of issues that still need to be addressed.

2.2 Wireless Sensor Networks

In the recent past, Internet-of-Things (IoT) deployments were confined to small implementations in personal houses or in laboratories; each individual creates their own Wireless Sensor Networks (WSN). Today IoT deployments are all over the world and in many application areas, such as Smart Cities, Intelligent Transportation Systems, Industry 4.0, etc.

These deployments are composed by WSN which are distributed systems formed by many battery-powered devices (called sensor nodes or motes), which can be actuators, gateways or distinct autonomous sensors that monitor physical or environmental conditions, such as temperature, sound, pressure, etc. (see Figure 2.2).

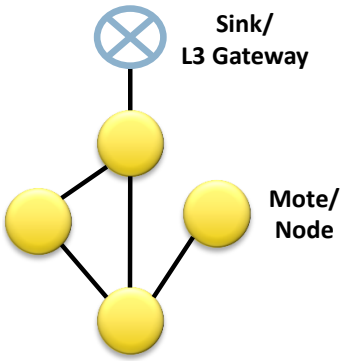


Figure 2.2: Example of a Wireless Sensor Network (WSN).

2.2.1 WSN Specific Issues

Sensor nodes (motes) present features from both embedded and general-purpose systems [38]. Due to their small size, limited resources, and other aspects, restrictions still persistent regarding computation, communication, power, etc. They are normally deployed in areas (e.g. a city, an office) using an ad-hoc fashion way (a need to-do base), which in some cases could result in situations where it is difficult to perform human interventions, such as, the constant replacement of node's batteries [39].

IoT Systems (motes) carry out simple and small applications which are developed using a specific sensor node Operating System (OS). In a sensor network, motes execute tasks such as sensing the environment (e.g. through temperature, humidity, CO2 emissions sensors), converting the analogical data into digital, send and receive data from/to the network using wireless links. The OS thematic will be addressed in Section 2.3.4.

Wireless Sensor Networks (WSN) presents some specific design and resource constraints, not seen in traditional networks. The resource constraints include the amount of available energy, short communication range, low bandwidth, limited processing and memory (storage) in each sensor node. Design constraints of a WSN are mainly the energy efficiency, cost and application requirements. An optimization at both hardware and software levels are needed to make a WSN more efficient [11].

IoT Systems operate typically in different cycles, namely snoozing (low power mode), processing, and transmitting data [39], since they are normally battery-powered devices, an efficient energy consumption policy is mandatory to increase their lifetime. Actually, the radio transmission is the operation that consumes more energy. The transmission of a single bit consumes about as much energy as executing 800 to 1000 instructions [40].

Nevertheless, the total energy consumption of a node is given by the addition of the energy spent by each physical component. Components, such as: radio, processor, sensor, leds and external memory. Each component consumes a specific amount of current to operate. Also the current spent by a component, depends from the state is operating (generally supplied by the manufacturer) [39]. Therefore, the total energy consumed by a physical component is given by the sum of the current spent at each one of the operating

states.

Unfortunately, there is no standard benchmark for WSN although some research groups have proposed some ideas as in [41] and [42]. This is mostly due to the fact that, without running the same application on each system is difficult, if not impossible, to compare fairly the performance of different systems. For example, a common used metric to compare different systems is the energy per instruction, but has being questioned because the notion of an instruction is lost when referring to architectures like Charm or Harvard Event-Driven (accelerator-based architectures). Therefore, the analysis through this metric could actually lead to completely misleading conclusions [10].

2.2.2 Resource-Constrained Systems (RCS)

With the growing interest of major technology players on Internet-of-Things (IoT), it has been seen a constant evolution of micro-electronic technologies, bringing to this domain a series of devices which enable the realisation of the IoT concept. Markets are offering a wide device diversity, with a very specific characteristic — Resource-Constrained. Manufacturers are engaged in developing new embedded systems for different purposes to address the variety of application domains and services. This factor enhances even more the established heterogeneous nature of IoT Deployments, and unlocks the use of a wide set of hardware platforms and software solutions (OS as implementation techniques).

Devices, also known as motes or sensor nodes, are categorised as Resource-Constrained Systems (RCS) as result of their particular features, such as small size, low-power, low processing and storage capabilities, sensing and wireless communication. However, another particularity emerges from such characteristics — devices Operating Systems (OS). IoT Systems tend to execute embedded applications such as sensing/monitoring the environment, act, along with some computation and communication tasks. Applications running on top of limited resources which needs specific OSs to be properly managed.

These characteristics allow Wireless Sensor Networks (WSN) to differ from other wireless technologies. It is a network of spatially disperse, distinct, autonomous and dedicated sensors, sensing physical or environmental conditions, with capability to collect and transmit data to a central unit.

Manufacturers are being able to offer reasonably cheap sensors, and still maintain accurateness/preciseness and reliability. A single device, it is by its own a system, composed by different hardware components interacting among them and running one or more programs/firmware, i.e. an IoT System. The most popular OSs for WSN will be addressed in Section 2.3.4.4.

2.2.2.1 Hardware Technology

Hardware technology applied in sensor nodes manufacturing is experiencing many changes due to the advances in Micro-Electro Mechanical System (MEMS), wireless communications and digital electronics that have led to small and cheap sensor nodes [11, 12]. A

wireless sensor node is composed by a micro-controller, memory, timer, transceiver, battery, sensing unit and Analog-to-Digital Converter (ADC) [43].

Figure 2.3 presents a block diagram of a typical architecture for a sensor node, which is composed by a set of hardware components, described as follows: A low frequency micro-controller when compared to traditional processing units (e.g. Personal Computers, Smart-Phones); with internal Random-Access Memory and Read-Only Memory modules; Several clocks to local synchronisation; External memory (flash memory) with better capacities than the internal memories; A transceiver enables wireless communication and supports WSN specific communication properties, such as low energy, low data rate and short distance communication; A battery providing energy; and finally an ADC to convert analogue data from the sensing unit (e.g. temperature, light sensor) to digital data.

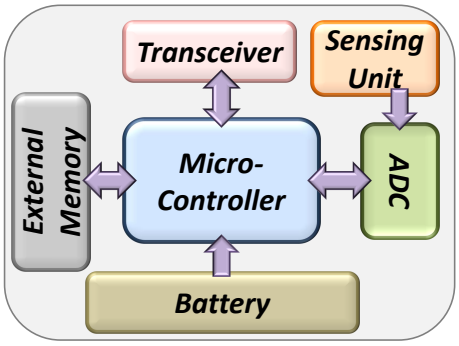


Figure 2.3: Typical Sensor Node Architecture.

Sensor nodes are typically design around its core module, the micro-controller. Some examples of micro-controllers are the Atmel ATMega 128L [44] and the Texas Instruments (TI) MSP430 [45], which are designed for low-power operations and general-purpose application. Although, these processor units support low-power idle states (consuming less than 5 μA in the case of the MSP430) that implies disabling the entire processor and waking it up on the next interrupt [10].

In a WSN is mandatory that sensor nodes own a transceiver (radio component) unit to connect them to the rest of the wireless network. The most commonly used radio component for motes is the TI-Chipcon CC2420 (Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee^{TM} Ready RF Transceiver) [46].

Motes are limited battery-powered devices, making the power unit one of the most important components in the sensor node architecture. There is significant research focus around making an energy-efficient management of these devices, namely harvesting and minimizing the energy consumption.

Energy harvesting (energy scavenging) is the process of retrieving energy from an external source. Known external energy sources are solar cells [47], vibration [48], fuel cells, acoustic noise, and a thermal diffusion [49]. Minimize the energy consumption can be achieved through significantly, by improving the energy performance of the applications.

2.3 Standards

The Internet-of-Things IoT is becoming an enormous success in terms of connecting people and all kind of objects. Objects have their own identifiers, with the ability to address other objects and verify their identities. Although, the lack of accepted standards and the huge fragmentation of the IoT market [50], makes the need for standardisation a key aspect in IoT. Many international organizations are working on define standards at different levels for the IoT (see Figure 2.4).

Figure 2.4: IoT Standardisation.

Standardisation initiatives face a wide group of available standards, each one stating that they are the solution to the problem in hand in their working area. Looking to IoT protocols, diversified standards are ready to use, such as Zigbee (802.15.4) or Bluetooth among others, from a communication point of view. The same happens at data interchange level. Standards like Advanced Message Queuing Protocol (AMQP) or Message Queuing Telemetry Transport (MQTT) are not unique but widely used by different platforms/middleware. Different platforms bring different OSs and software.

The following sections point out the more important organisations and alliances working to grasp and solve this issue in IoT, that is *"standardisation"*. The large scope of protocols, platforms and consequently software is also addressed.

2.3.1 Organisations & Alliances

The European Telecommunications Standards Institute (ETSI)[2] is a leading standardisation organisation, that produces high quality, innovative and globally applicable standards for the ICT. Committed to standardise fixed, mobile, radio, converged, broadcast and internet technologies, protocol testing and methodology and they also offer forum-hosting services. ETSI is an independent, not-for-profit organisation with more than 700 member organisations over 62 countries across 5 continents world-wide, and is officially recognized by the European Union as a European Standards Organization.

The Internet-of-Things (IoT) is becoming an enormous success in terms of connecting people and all kind of objects, since in a not too distant future, objects will have its own identifier, the ability to address other objects and verify their identities. According to ETSI, "An increasing number of everyday machines and objects are now embedded with sensors or actuators and have the ability to communicate over the Internet. Collectively they make up the Internet-of-Things (IoT)".

ETSI is committed to standardise various technologies, such as Radio-Frequency identification (RFID), Machine-to-Machine (M2M) and WSN to integrate these 'smart objects' into a communication network and guarantee its openness and interoperability. ETSI is one of the members of oneM2M, a global partnership initiative to provide a standard M2M interface, to enable different devices to connect, independently of the network, to the IoT.

International Organization for Standardization (ISO)[3] is a non-governmental, independent international organisation focus on sharing knowledge and develop international standards, relevant to the market, that supports innovation and bring solutions to world-wide challenges. Together with International Electrotechnical Commission (IEC) they have been providing standards in IoT scope, such as: an interoperability framework for information exchange between IoT Ecosystems and use that information in an efficient way; design of a standard reference-architecture for IoT; description of concepts, characteristics, and technologies for edge computing; as many other. At this moment, ISO and IEC are working together to present standards for Cybersecurity — IoT security and privacy.

The Institute of Electrical and Electronics Engineers (IEEE)[4] created an initiative call IEEE IoT Initiative with the objective of bring the IoT global technical community together, providing a platform for professionals where they can learn and share knowledge, collaborate on technologies, applications and markets surrounding the IoT. The IEEE IoT is one of IEEE's important, multi-disciplinary, cross-platform Initiatives.

The Internet Engineering Task Force (IETF)[5] is a large open international community

[2] https://www.etsi.org/
[3] https://www.iso.org/
[4] https://www.ieee.org/

concerned with the evolution of the Internet architecture and its Internet smooth operation. To do so, IETF, focus in produce relevant, high quality technical documents that can influence people on their design, use and manage of the Internet.

IETF working groups, in multiple areas, develop standards that are directly relevant to the IoT. Some working groups are: the Constrained RESTful Environments (CoRE) Link Format which aims to extend the Web architecture to most constrained networks (e.g., IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) [RFC4919]) and embedded devices (e.g., 8-bit microcontrollers with limited memory); the 6LoWPAN group which defined mechanisms to encapsulate and compress IPv6 packets headers, so they could be used over IEEE 802.15.4 based networks, i.e. in constrained radio links;

The International Telecommunication Union (ITU)[6], founded in 1865, is the oldest specialized agency in United Nations (UN). It is responsible for information and communication technologies issues. Namely, coordinate the use of radio spectrum globally; responsible for international cooperation regarding satellite orbits; improve world telecommunication infrastructure; and assist in the development and coordination of technical standards worldwide. ITU includes three sectors, besides ITU Telecom, which are the Radio communication (ITU-R), Development (ITU-D) and Standardization (ITU-T).

The ITU Telecommunication Standardization Sector (ITU-T) are study groups formed by experts around the world, with the goal of develop international standards to be used by the global infrastructure of ICTs. These standards are known as ITU-T Recommendations. One example of ITU-T study groups is the *"SG20 - IoT, smart cities & communities"*.

The Object Management Group (OMG)[7] is an international, open membership, not-for-profit technology standards consortium, founded in 1989. Its work is focused on the development of integration standards for technologies and industries. OMG provides modelling standards, such as Unified Modelling Language (UML) and Model-Driven Architecture (MDA), which facilitates software design through visual blocks integration, as well as, development and maintenance of software and other process. OMG presents also a middleware standard specifically for real-time and embedded systems called OMG Data Distribution Service (DDS) for Real Time Systems, using publish-subscribe communications.

The World Wide Web Consortium (W3C)[8] is an international community working together to develop Web standards. W3C goal is to join different interested Information Technology (IT) parties to work on web issues: *"The objective of the W3C is to bring the web to its full potential"*. As the primary international standards organisation for the web, the consortium includes well know standards like Hypertext Transfer Protocol (HTTP), Hypertext Markup Language (HTML) and Extensible Markup Language (XML).

W3C created the Web of Things Community Group to easy the adoption and development of standards for the Web of Things, counter the fragmentation of IoT, enabling

[5]https://www.ietf.org/
[6]https://www.itu.int
[7]http://omg.org/

services and devices for the IoT. One W3C Recommendations published regarding the IoT is the Semantic Sensor Network (SSN) ontology, intended to describe/model sensors, systems, processes and observations.

The Organization for the Advancement of Structured Information Standards (OASIS)[9] is a global not-for-profit consortium originally founded in 1993 under the name "Standard Generalized Markup Language (SGML) Open", changed in 1998. OASIS focus on the development, convergence and adoption of open standards in many areas, like IoT, security, energy, cloud computing, etc. Some of the well-known standards proposed by OASIS are for example AMQP and MQTT.

As stated previously, oneM2M[10] is a global partnership initiative to provide a standard M2M interface, to enable different devices to connect, independently of the network, to the IoT. It was launched at July 24, 2012 by seven of the world's leading ICT Standards Development Organizations (SDOs) to ensure the most efficient deployment of M2M communication systems. oneM2M members are working in technical specifications to provide a common platform to support services and applications in different areas, such as, smart grid, connected car, eHealth, etc. Ultimately, the work developed will drive multiple industries to a lower operating and capital expenses, shorter time-to-market, expand and accelerate global business opportunities, and avoiding standardisation overlap.

Dozens of organisation and coalitions have been formed with the objective of unify the fragmented, fractured and organic IoT landscape, although others could be point out. Namely, the Open Geospatial Consortium (OGC)[11], an international voluntary consensus standards organization that is also a member of the ITU-T; International Society of Automation (ISA)[12] a not-for-profit professional association that sets engineering and technological standards in management, safety and cybersecurity areas; International Electrotechnical Commission (IEC)[13] an international standards organisation working for all electrical, electronic and related technologies; etc.

2.3.2 Protocols

When it comes to IoT protocols, developers encounter dozens of competing communication protocols. Next, is given an overview of IoT protocols divided in two main topics: Wireless Communication Protocols; and Data Protocols. The first, focus on physical transmission, while the second on how messages are built and exchange.

[8] https://www.w3.org/
[9] https://www.oasis-open.org/
[10] http://www.onem2m.org/
[11] http://www.opengeospatial.org/
[12] https://www.isa.org
[13] https://www.iec.ch/

2.3.2.1 Wireless Communication Protocols

Wireless communication protocols can be divided in groups, either by transmission range, frequency or by the amount of data transmitted (bit rate). The standard list is immense, therefore, only a few are addressed in detail. Protocols are organised by transmission range. The selected protocols are RFID, Near-field communication (NFC), Zigbee, Z-Wave, Bluetooth, Wi-Fi, WirelessHart, SigFox, Long-Range (LoRa), Narrowband IoT (NB-IoT) and Global Positioning System (GPS). Although, there are others such as WiMax, EnOcean, ANT, etc.

Usually the communication protocols in IoT are divided into 3, 4 main groups regarding transmission range. For very close transmission is designated the Wireless Personal Area Network (WPAN), that includes transmissions up to 10 meters long. The Wireless Local Area Network (WLAN) covers transmission areas up to 1 kilometre. Wireless Metropolitan Area Network (WMAN) that goes up to 50 kilometres. Last one, Wireless Wide Area Network (WWAN), that in some literatures aggregates WMAN, covers all higher transmission ranges. See Figure 2.5.

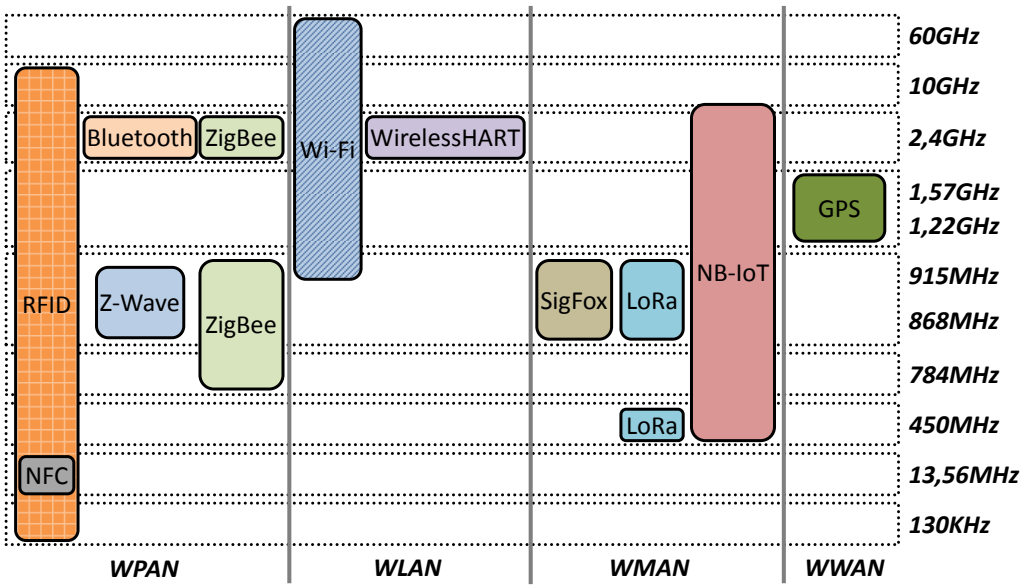


Figure 2.5: IoT Protocols: Radio Frequency and Network Areas.

WPAN - Wireless Personal Area Network

Radio-Frequency identification (RFID) is the process where items are identified using radio waves. At a minimum it is formed by a two nodes network, a passive device (e.g.: a tag) and one active device (powered). Tags contain electronically-stored information which retrieves power from radio waves emitted from a near active device, making it readable. Data exchange between two nodes is also possible if both devices are powered. RFID is used in different frequencies, that can go from 120 KHz (e.g.: animal identification) to 3.1 – 10 GHz (needs active tags). Lower frequencies allow transmission ranges around 10 centimetres going up to 200 meters with higher frequencies [51].

The NFC is a communication protocol that uses electromagnetic induction between two antennas (two devices) [52], based on the RFID standard. NFC operates on 13.56 MHz radio frequency, which is globally available but unlicensed [53]. Works on a very low distance between devices, normally 4 centimetres, with a transmission rate from 106 to 424 kilobit per second (kbit/s). A active-passive network is used to identify objects or retrieved PINs (e.g.: for payments).

Zigbee is a wireless communication based on the IEEE 802.15.4 specification for WPAN. Zigbee works on the 2.4GHz radio band, i.e. Industrial, Scientific, and Medical (ISM) band. Although, in some countries it can operate at different frequencies, like in China at 784 MHz, 868 MHz in Europe and at 915 in USA and Australia. Data transmission can go from 10 – 20 meters indoor, up to 1500 meters outdoor depending on power output and environmental characteristics [54]. Regarding transmission rate, in the 868 MHz band it is 20 kilobit per second (kbit/s), and 250 kbit/s in the 2.4 GHz band.

Z-wave had its physical and media access control layers included as an option in the ITU *G*.9959 standard for wireless devices under 1 GHz. Designed for low-latency transmission of small packets with a data rate up to 100 kbit/s [55]. It operates at 908.42 MHz in North America and at 868.42 MHz in Europe. Communications range between two nodes it of 30 – 40 meters with capability to hop between four nodes.

Bluetooth is built mostly for point-to-point wireless communication, exchanging data over short distances. Although, it works also in broadcast and mesh topologies. Bluetooth Class 3 can send data up to 1 metre, Class 2 the more common can go up to 10 meters, and Class 1 used in industrial scenarios can go up to 100 meters [56]. Operates in 2.4 to 2.485 GHz ISM radio band, with data rates that can go from 125 kbit/s to 2 megabit per second (Mbit/s) for the Bluetooth Low-Energy (LE), and up to 3 Mbit/s with Enhanced Data Rate (EDR).

WLAN - Wireless Local Area Network

Wi-Fi is the wireless communication technology more used in the entire World [57]. Its technology is based on the IEEE 802.11 standards, with different radio bands, in which 2.4 GHz and 5 GHz are most common. Transmission rates can go up to 600 megabits per second (Mbit/s), 6.9 gigabit per second (Gbit/s) or 9.6 Gbit/s with versions 4 (802.11n), version 5 (802.11ac) and version 6 (802.11ax) respectively. Transmission ranges can vary depending on environmental conditions: indoor/outdoor conditions; clear sight; no reflecting objects.

WirelessHART is a wireless protocol based on the Highway Addressable Remote Transducer Protocol (HART). Operates in the license-free ISM radio band of 2.4 – 2.483 GHz, like IEEE 802.15.4. It was developed as an open industrial standard to tackle the need for wireless communication at field level in industrial process [58]. WirelessHART transmission rate can go up to 250 kbit/s, with a range around one hundred meters.

WMAN - Wireless Metropolitan Area Network

Sigfox, like other communication protocols already presented, uses two frequencies from the ISM radio bands, 868 MHz for Europe and 902 MHz for United States. Sigfox

radio messages are small, less than 26 bytes, but with a limited number of messages per day. Cells are deployed in 30 – 50 kilometres average range in rural areas, while in urban areas it is 3 – 10 kilometres apart. Transmission goes from 100 to 600 bits per second [59].

LoRa (Long-Range) uses license-free frequency bands depending on the place it is used. In Europe, South Africa, Asia and Australia it is used 433 MHz. 868 MHz frequency is used in Europe. United States and Australia use a 915 MHz band. LoRa end nodes can transmit over 5 – 10 kilometres range. LoRa data rates goes from 290 bit/s to 50 kbit/s depending on the frequency band used [60, 61].

The NB-IoT is a cellular IoT technology working in a licensed spectrum (Long-Term Evolution (LTE), GSM and UMTS bands). Also known as LTE Cat NB1, NB-IoT is a part of the 3GPP Release 13 standard. NB-IoT can operate in three modes: inband, guardband, or standalone. Due to NB-IoT small bandwidth size, 200 KHz, it can be deployed in side LTE band — inband, between two continuous LTE bands — guardband, or it can be deployed between two GSM channels — standalone. NB-IoT provides a communication range up to 15 kilometres with a peak rate up to 250 kbit/s for downlink and uplink transmissions [62–64].

WWAN - Wireless Wide Area Network

The Global Positioning System GPS is a satellite system owned by the United States government, than transmit radio signals for position, navigation and time services. With a network of at least 24 operational satellites, messages are continuously broadcast to GPS receivers on two different frequencies. One at 1.57542 GHz and the second at 1.2276 GHz with a 50 bit/s transmission rate. Satellites fly in medium Earth orbit at an altitude of approximately 20.200 kilometres [65].

Table 2.1 shows the wireless protocols mentioned previously organised by transmission rate.

Table 2.1: IoT Protocols: Transmission Rate.

50 bps	**100 bps**	**290 bps**	**600 bps**	**50 Kbps**	**100 Kbps**	**125 Kbps**	**250 Kbps**	**424 Kbps**	**2 Mbps**	**3 Mbps**	**600 Mbps**	**6.9 Gbps**	**9.6 Gbps**
GPS	*Sigfox*				*Z-Wave*	*Bluetooth (BT) Class 1*	*NB-IoT; WirelessHart; Zigbee*	*NFC*	*BT Class 2*	*BT Class 3*	*Wi-Fi*		
		LoRa											

2.3.2.2 Data Protocols

Data elements are exchange between components to trigger a certain procedure or just to transfer data. The use of asynchronous communication makes the relationship between components less narrow when comparing to the common synchronous method [66, 67].

Asynchronous communication allow systems to work more freely, individual subsystems communicate with no restraints on tasks execution speed, clock rates or network delays. Independently, subsystems continue with their functions/tasks. There is no blocking stage waiting for another subsystem response [68]. In this sense, the use of asynchronous communication provides scalability and extensibility to systems [67]. Subsystems can go to sleep, wake up or new ones can be added to the global system that it will not break. Also, global systems can take advantage by including other subsystems, capabilities and functionalities can be removed, increased or create new ones.

Asynchronous communication is present in our daily life, for example when a email is sent, is used a send-and-forget approach. Or when visiting a web site, the browsers will not freeze waiting for a response allowing users to perform other tasks. However, not always an asynchronous solution can be applied, for example, a real-time communication between two persons. In these cases, a synchronous communication must be used.

Message exchange, i.e. interaction, between nodes/components is based in mechanisms often called Message-Oriented Middleware (MOM). MOM is a specific software class that gives support for message exchange among a distributed environment of software and hardware systems [69]. Based on this approach, several application layer protocols were developed, for example: AMQP, MQTT, Constrained Application Protocol (CoAP), Streaming Text Oriented Messaging Protocol (STOMP), DDS, Extensible Messaging and Presence Protocol (XMPP) and Java Message Service (JMS).

AMQP was born in 2003 by the hand of Jonh O'Hara to tackle the development of front- and back-office processing systems for investment banks. AMQP is built upon two main models, the transport and queuing model [70]. The transport model is based in a binary protocol that uses network byte ordering, to address high performance and flexibility, and therefore is more hardware friendly than human friendly. Queuing model deals with services semantics to achieve interoperability among entities as well as storage. Queues reside on final destinations or intermediate components.

To support message delivery, AMQP provides the use of Point-to-Point (PtoP) and store-and-forward message exchange. Also, messages size can go up to several gigabytes.

Due to demand and general use, several known AMQP implementations have emerged. From proprietary: IBM WebSphere MQ or Microsoft Message Queuing (MSMQ); to Open-Source Software (OSS): RabbitMQ[14], Qpid[15], or ActiveMQ[16]; to mention some examples.

Another application layer protocol is the Message Queuing Telemetry Transport (MQTT) [71], which is an OASIS Standard (currently at version 3.1.1) for message transport. Developed by Dr Andy Stanford-Clark and Arlen Nipper of Arcom in 1999, MQTT stands for MQ Telemetry Transport. It is a M2M (or Client Server) publish-subscribe message protocol, built to be simple, open, lightweight and easy to implement a connection protocol for the IoT. Consequently, ideal for constrained environments where small size,

[14] http://www.rabbitmq.com/
[15] http://qpid.apache.org/
[16] http://activemq.apache.org/

low power, small number of data packets, and small code footprint are important requirements. MQTT protocol runs over a network protocol that provides ordered, lossless, bidirectional connections, for example: Transmission Control Protocol / Internet Protocol (TCP/IP).

Another application layer protocol is the Constrained Application Protocol (CoAP) [72], specified as RFC 7252 in the Standards-Track by IETF. CoAP was designed for M2M applications, in which the application interaction follows the request/response model and with capability to discover services and resources. The main objective of CoAP protocol is to be a generic web protocol focus on the specific characteristics of a constrained environment (nodes and networks). These characteristics includes low-power, low memory (ROM and RAM), multicast support, very low overhead. It also presents an easily interface with HTTP, for integration with the Internet.

Streaming (or Simple) Text Oriented Messaging Protocol (STOMP)[17], is a wire format interoperable protocol design for asynchronous message exchange. Similar to HTTP, it distinguishes itself from others by using only a few message operations rather than provide a complete messaging Application Programming Interface (API). It has been used for many years, supported in several message brokers and client libraries (as demonstrated previously), STOMP is an alternative protocol to others, like AMQP and JMS brokers specific implementations (e.g. OpenWire). STOMP allows servers to define what semantics they want to use, changing from server to server, as well as, from client to client, making it a more flexible protocol [73].

Data Distribution Service (DDS) [74, 75], is a data communication standard for publish-subscribe systems managed by the OMG[6]. DDS specification provides a distributed application communication and integration, using a Data-Centric Publish-Subscribe model. This model is based on the principle of a "global data space", in which data is also available between time-decoupled publishers and consumers.

OMG DDS M2M middleware intends to be scalable, secure and with a high-performance. It is a complete decentralised architecture, providing a dynamic discovery service, Quality-of-Service (QoS), security, web integration, among others characteristics. DDS is independent of language and OS, and has been implemented in C, C++, Java, JavaScript, etc.

Extensible Messaging and Presence Protocol (XMPP)[18] [83] is a communication protocol to exchange messages and presence information using XML streams. Originally known as Jabber (created by Jeremie Miller in 1998), it was developed mainly for instant messaging, chats, voice and video calls, collaboration, etc. The Jabber open source community wanted to provide an open and decentralized alternative to the existent closed instant messaging. In 2004 the XMPP core functionality was published as RFC 3920 and RFC 3921 by the IETF. Jabber Software Foundation renames itself to XMPP Standards

[17] http://stomp.github.io/

Table 2.2: Message-Oriented Middleware (MOM) Protocols Summary.

MOM Protocol:	Paradigm:	Implementation Architecture:	Discovery:	Transport:	Quality of Service:	Security:	Encoding:	Open-sources:	References:
AMQP	PtoP or Publish-Subscribe	Broker	No	TCP/IP	Up to 3 parameters	SASL and/or TLS	Binary	ActiveMQ, RabbitMQ, OpenAMQ, Apache's Qpid	[69][70] [76][77]
MQTT	Publish-Subscribe	Broker	No	TCP/IP	Up to 3 parameters	TLS	Binary	ActiveMQ, RabbitMQ, Mosquitto	[69][71] [76][77]
CoAP	Request-Reply (Representational State Transfer (REST))	Broker or Broker (Server)	Yes	UDP/IP	Up to 2 parameters	DTLS and IPSec	Binary	RabbitMQ	[78][79] [80][81]
STOMP	PtoP or Publish-Subscribe	Broker (Server)	No	TCP/IP	Application Dependent	Application Dependent	Text-based or Binary	ActiveMQ, RabbitMQ, HornetQ	[73][76]
DDS	Publish-Subscribe or Request-Reply	Global Data Space	Yes	UDP/IP or TCP/IP	Up to 22 parameters	TLS, DTLS, DDS Security	Binary	OpenDDS, Vortex OpenSplice	[74][75] [77][82]
XMPP	PtoP or Publish-Subscribe	Broker (Server)	Yes	TCP/IP	None	TLS and SASL	Plain Text	ActiveMQ	[83][76] [77]
JMS	PtoP or Publish-Subscribe	Broker (Server)	No	TCP/IP	Up to 3 parameters	SSL or TLS	Binary	ActiveMQ, HornetQ, Open MQ, Apache's Qpid	[84][85] [76][77]

Foundation — XSF in 2007, and starts focusing on the development of XMPP specification, instead of open source software development.

XMPP is an open standard protocol, with a decentralised architecture capable of providing secure, extensible and flexible message exchange. It is platform independent with support for Windows, Linux, Android, OSX, etc. with a variety of programming languages implementations like C, C++, Java, JavaScript, PHP, Erlang, Perl, etc. XMPP is used by well-known companies such as: Google, Whatsapp and Apple.

Finally, Java Message Service (JMS) [84], currently at version 2.0 of its specification, is used by Java applications to send, receive and read messages. JMS supports both PtoP and Publish-Subscribe messaging allowing components communication to be loosely coupled, reliable and asynchronous. However, applications in a JMS communication must connect to a JMS server. A few known MOM implementations, which support JMS specifications are: ActiveMQ, HornetQ, and Apache Qpid (a JMS client API).

Previously were presented, for the application layer, well known and more relevant protocols. Namely, the AMQP [70], MQTT [71], CoAP [72], STOMP [73], DDS [74], XMPP [83] and JMS [84]. Table 2.2 shows an analysis of the mentioned MOM standard protocols, considering some key criteria. On the other hand, Figure 2.6 presents each protocol accordingly to the connectivity space it addresses.

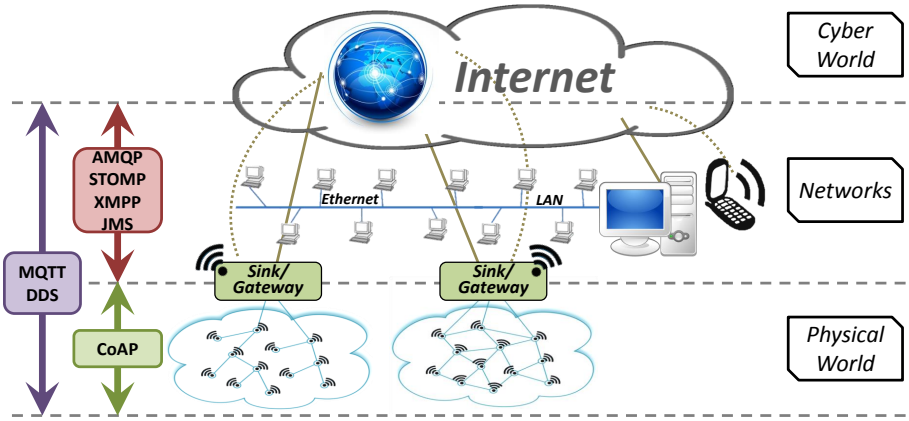


Figure 2.6: Connectivity Space of MOM Protocols.

Analysing the MOM protocols presented here in terms of open source implementation, stands out three open source message brokers. These open source middleware implementations are the RabbitMQ, Apache Qpid and ActiveMQ (see Figure 2.7). RabbitMQ [86] had become one of the most popular implementation of AMQP. It is an open source, lightweight, reliable, scalable and portable message broker, written in Erlang, easy to use and suitable for a cloud scale. It accepts connections through multiple platforms (e.g.: Windows Servers, Linux, Mac OS X), capable of using different message protocols, such as MQTT, Simple Mail Transfer Protocol (SMTP), STOMP or HTTP [87]. RabbitMQ provides client libraries in other programming environments, like Java, Python, C/C++,

[18] http://xmpp.org/

etc. Its major drawbacks are a poor server's clustering, no scalability support for more than 250.000 clients and queues, and it does not react well to DNS/DHCP failures.

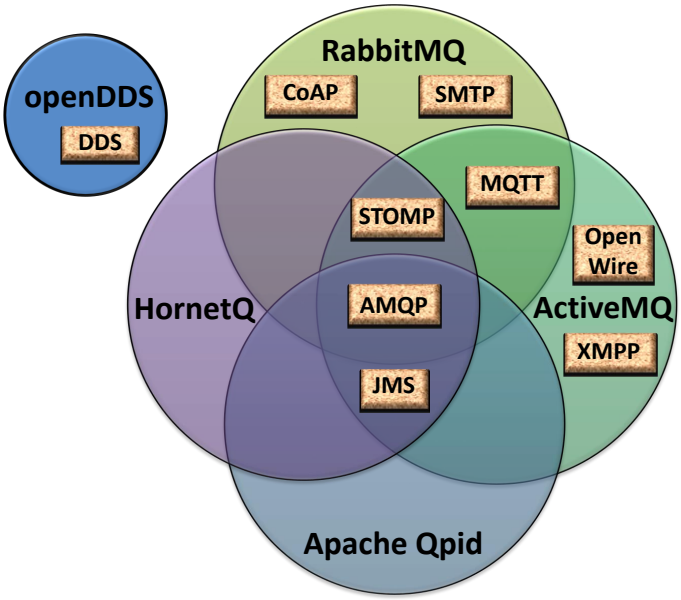


Figure 2.7: MOM Protocols vs Open Source Message Brokers.

Apache Qpid is a distributed priority queue algorithm for messages passing. Items are inserted in a logical distributed queue, to be later on removed and processed using a priority method [88]. Qpid is composed by two types of components, the messaging API and messaging servers. APIs are a tool used for high-level communication, and to manage messages acknowledges. They are implemented using for example Java, C/C++, Perl, Python, among others. Messaging servers, implemented using Java and C++, are messages brokers working as store-and-forward entities, ensuring messages delivery even if destination is disconnected. Working environment includes Linux, Windows and Java Virtual Machine.

ActiveMQ is an open source message broker, which belongs to the Apache Software Foundation. It is built using Java and uses a full JMS client implementation. Messages are sent asynchronous or synchronous by a producer to the message broker, which will transmit the message to the destination (consumer). Consumer confirms the message reception to the broker [89]. ActiveMQ has been tested in the Windows, Mac OS X and Linux platforms. It supports several message protocols for a maximum interoperability, such as: AMQP, OpenWire (binary format), STOMP, MQTT, XMPP, etc., which enables the use of clients implemented in C, C++, Python, PHP, etc.

Another aspect regarding IoT data is data semantic. Data semantics is associated to what the data means, to what it concerns for humans rather than to machines interpretation. In a data exchange, within a WSN, there must be a common understanding between "things" of the meaning of the content (information) being exchanged. The use

of semantic technologies, in particular ontologies, for domain modelling and knowledge representation has been seen as a promising approach to address the distributed and heterogeneous nature of the IoT, in a way that can facilitate automated machine processing [6].

Semantic web based standards from W3C, like DAML (DARPA Agent Markup Language), RDF (Resource Description Framework) and OWL (Ontology Working Language), are useful in providing semantic foundations for dynamic situations involving dynamic discovery of devices and services, and to encompass all contexts of information, so that devices can interpret information from other devices.

2.3.3 Platforms

Execute an IoT deployment from tail to head is a slow and complex work. Any assistance is welcome and IoT data platforms already have the necessary tools for connectivity and data communication with devices, as well as, services management. Next is presented some IoT data platforms examples covering 3rd party companies, open source and Do it yourself (DiY) enablers.

2.3.3.1 Supported by Companies

The first example of an IoT data platform provided by 3rd party companies is the Autodesk[19] Fusion Connect. Fusion Connect is a cloud based IoT application platform focus on two topics, scalable and applications building. Collects data from physical devices thorough a cloud native connection interface, called Device Adapter Layer, independently of message protocols (e.g.: MQTT, CoAP, HTTP). Messages are parsed using a large standard library and custom device adapters. Applications building (develop an IoT App) is provided by a no-coding development environment with all necessary tools to construct a fully functional application [90].

Amazon Web Services (AWS)[20] IoT is a platform that provides secure and bi-directional communication between devices (e.g.: sensors, embedded micro-controllers), smart appliances and AWS cloud. With capability to connect billions of devices, transport trillions of messages and applications can track devices all the time, even when they are disconnected. Connectivity is supported by HTTP, WebSockets and MQTT. Also, other industry-standards and custom protocols are supported. Services include gathering, process, analysis, as well as, act upon the generated data retrieved from devices, without the need to manage any infrastructure [91].

The General Electric's (GE)[21] Predix platform is a distributed application and services platform to build and run digital industrial solutions. Predix platform was built to tackle connectivity and security issues in data exchange with remote assets, common in industrial severe environments. Based on a cloud and edge approach, Predix platform

[19] https://www.autodesk.com/
[20] https://aws.amazon.com/

technology enables data processing, storage and data analysis closer to machines leading to a tight union between machines, control systems, and modern applications, as well as, smaller amount of data that needs to travel to Predix services. Connectivity between assets is achieved through a bi-directional communication over OPC-UA, Modbus, or MQTT protocols [92].

Another example of an IoT data platform provided by 3rd party companies is the Google Cloud IoT. Similar to the GE Predix platform, Google Cloud IoT provides tools to connect, process, store and analyse data in the cloud as at the edge (Cloud IoT Edge). Google is responsible for the infrastructure, meaning that Google Cloud IoT services are fully manage and do not need user's local storage. Devices and gateways are connected using MQTT and HTTP standard protocols, from which all data captured is published, using a publish/subscribe service, to be consumed by analytic services [93, 94].

Microsoft Azure IoT Suite is a platform composed by services that enables users to interact (connect, monitor, and control) with billions of IoT devices. Azure IoT services include management of industrial equipment, healthcare tracking assets, and monitoring building usage. Connections with IoT devices are made through IoT Hub, a managed service hosted in the cloud, scalable to millions of simultaneously connected devices and millions of events. Communication takes place using standard protocol such as Hypertext Transfer Protocol Secure (HTTPS), AMQP and MQTT. In case, the devices are not able to implement such protocols, native connections can be established to the hub using HTTPS $v1.1$, AMQP $v1.0$ or MQTT $v3.1.1$ [95, 96].

A sixth example is the IBM Watson IoT platform. A fully managed platform, where services are hosted in the cloud, to make it more easy to generate value from IoT devices. Features such as device registration, secure connectivity, control, fast visualisation and data storage enables organisations to explore new services and perform better decision-making. Devices connectivity is achieved by open standards-based communications like MQTT and HTTPS [97, 98].

Nevertheless, there are other 3rd party companies' platforms that could be mentioned. With the grouth of IoT market, companies are committed to create their own solutions focus on their point of view as on key aspects. Other IoT platform examples are ThingWorx[22] by PTC, Salesforce IoT Cloud[23] or WebNMS Platform[24], to mention a few.

2.3.3.2 Open Source Platforms

Kaa's open source IoT platform goes by the name of Kaa Community Edition. It is the first generation of the Kaa IoT Platform. Kaa Community Edition provides device management, data collection, configuration management, messaging, and more. Open source Kaa platform needs third-party components to work. Among these components stands out,

[21] https://www.ge.com/
[22] https://developer.thingworx.com/en/platform
[23] https://www.salesforce.com/products/salesforce-iot/overview/
[24] https://www.webnms.com/

PostgreSQL for object-relational database management; the MariaDB as database server; Zookeeper for services managing; and MongoDB or Cassandra as a NoSQL database [99]. Kaa Community Edition platform uses HTTP communication, while Kaa IoT Platform supports MQTT and CoAP protocols [100].

Another IoT open source platform is the SiteWhere Platform[25]. SiteWhere provides a platform for industrial open source IoT applications, with a variety of micro-services that scale independently and a easy platform integration. SiteWhere infrastructure is built upon Kubernetes[26], an open source container orchestration engine to manage workloads and services, allowing a local deployment or in any cloud provider. Available micro-services included assets or events management, big-data storage among others through a modern, scalable architecture [101]. Communication uses MQTT, by default, with the possibility of multiple encodings to allow interaction with different device types [102].

The last open source platform example to be presented is the ThingSpeak Platform[27]. ThingSpeak is a free service for non-commercial small projects, whit less than 3 million messages per year or around 8200 messages per day. ThingSpeak Platform provides services to aggregate, visualize and analyse instant data streams in the cloud posted by devices. ThingSpeak has the possibility to use MATLAB to perform online analysis and consequently make sense of the incoming IoT data. Communication between devices and ThingSpeak is made using TCP/IP, HTTP or MQTT protocols [103, 104].

Many other open source platforms could be point out, such as Macchina Platform[28] that offers the possibility of a free account with up to five devices.

2.3.3.3 Friendly Platforms for devices DiY

This section will describe IoT platforms that are focus on the use of well-known DiY hardware platforms, such as Arduino, Raspberry, etc. The use of such platforms smooths device integration, in deployment time as well in difficulty.

Altair SmartWorks[29] is a cloud-native platform to implement easy-to-use, reliable and highly-scalable IoT projects. A solution to lead to innovation based on advance edge-to-cloud IoT applications and augmented data analytics using machine learning. Altair SmartWorks presents as main features its flexibility, high compatibility with 3rd party hardware, communication technologies and applications. Communication is achieved using REST API and MQTT brokers.

Another example is the Ubidots Platform[30], an IoT data analytics and visualization company, focus in connect hardware and software solutions to remotely monitor, control, and automate processes delivering agile solutions that improve company's key performance indicators and/or services. Point-and-click tools are provided for self-building

[25] https://sitewhere.io/
[26] https://www.kubernetes.io/
[27] https://www.thingspeak.com/
[28] https://macchina.io/
[29] https://www.altairsmartworks.com/

applications, enabling developers to create applications that best fit their needs. Ubidots Platform has a time-series infrastructure optimised to deal with millions of data points per second across the globe.

Several examples, tutorials are available for both hardware and software implementation/integration to the Ubidots platform. Engineers encounter wide hardware solutions such as Arduino, Electrical Imp, Microchip, Raspberry Pi and many others. Communication is done over HTTP, Transmission Control Protocol (TCP), User Datagram Protocol (UDP) with MQTT or REST API. Parsing industrial/custom protocols is also possible [105].

There are other examples, such as: the Initial State Platform[31], that presents tutorials in how to use the Raspberry Pi, Arduino and Electric Imp hardware; the MyDevices Cayenne Platform[32] with Raspberry Pi, Arduino devices and other hardware platforms; and Temboo Platform[33] that offers solutions with Arduino and Arduino-compatible devices, Samsung ARTIK boards, and with Texas Instruments microcontrollers.

2.3.4 Embedded Operating Systems (OS)

Several Operating Systems (OSs) have being specifically design to meet the requirements of Wireless Sensor Network (WSN) applications. The main challenge is to efficiently manage the sensor nodes scarce hardware resources (memory, processor and power), but also allow multiple applications to use simultaneous the system resources (e.g.: communication, computation, memory) [106].

By being of very different nature, WSN OSs presents one of two execution models types, event-based or thread-based [39]. According to the authors in [43, 107] the most popular WSN OSs are: TinyOS, Contiki and MANTIS. Next an overview of each one will be presented.

2.3.4.1 TinyOS

The TinyOS [108] is a multi-platform, flexible, component based and open source WSN operating system. This OS presents a footprint of about 400 bytes. TinyOS can support concurrent programs, with low memory requirements and it falls under a monolithic architecture class. Its execution model is event-based.

TinyOS is written in Network Embedded Systems C (nesC) [109]. nesC is a component-based programming language based on C, that allows programming interfaces and components. TinyOS uses components as independent computational entities that expose one more interfaces. Components present three computational abstractions: commands, events and tasks. Commands and events are mechanisms for inter-component communication, while tasks are used to express intra-component concurrency.

[30] https://ubidots.com/
[31] https://www.initialstate.com/
[32] https://www.mydevices.com/
[33] https://www.temboo.com/

The earlier versions do not provide multithreading and the programming was done following an event driven model. Version 2.1 of the TinyOS introduce multithreading, called TOSThreads [110], coupled with an efficient event driven kernel. TOSThreads context switches and system calls introduce an overhead of 0.92% or less to computation (clock cycles) [108].

Previously it supported a non-pre-emptive First-In-First-Out (FIFO) scheduling algorithm, causing the unavailability for real-time applications. Consequently, TinyOS shared the disadvantages associated to the FIFO scheduling (waiting time depend of tasks execution time). Although, in [108] authors claim the additional support for Earliest Deadline First (EDF) scheduling. Nevertheless, EDF algorithm does not produce a feasible schedule when tasks concur for resources. Therefore, TinyOS does not provide a solid real-time scheduling algorithm.

TinyOS version 2.1.1 provides communication support for 6lowpan [111], an IPv6 networking layer within a TinyOS network, reserving the earlier versions protocols, Dynamic MANET On-demand (DYMO) and DIssemination Protocol (DIP) [112]. At the MAC layer, it provides implementation for the following protocols: a single hop TDMA protocol; a TDMA/CSMA hybrid protocol which implements Z-MAC's slot stealing optimization; B-MAC; and an optional implementation of an IEEE 802.15.4 complaint MAC.

To accomplish resource sharing, TinyOS uses two management mechanisms: the Virtualization where a resource is virtualize to provide independent instances of that resource; and the Completion Events to handle resources that are not possible to be virtualized. It also gives support to other features, such as, database support through the form of TinyDB [113], security for communications in the form of TinySec [114] and simulation through TOSSIM [115] (simulation code is written in nesC which can be directly deployed to nodes). Another important characteristic of TinyOS is that it is very well documented. Extensive documentation can be found at http://www.tinyos.net, the TinyOS home page.

2.3.4.2 Contiki

Contiki [116] is a lightweight and flexible OS for tiny networked sensors. It was developed in the Swedish Institute of Computer Science (SICS) by Adam Dunkels as project leader in 2003. Contiki is an open source OS, written in C [117] and a typical installation consumes 2 kilobytes of RAM and 40 kilobytes of ROM. Contiki OS follows a modular architecture. At the kernel level it uses a lightweight event scheduler or polling mechanism to trigger processes. The events can be of two types: asynchronous or synchronous events.

On top of the event-driven kernel, Contiki supports the implementation of pre-emptive multithreading using protothreads [118], which are design for high memory constraints. Since events run to completion, and Contiki does not allow interrupt handlers to post new events, no process synchronization is provided [43]. Also being an event-driven OS, it does not provide any sophisticated scheduling algorithm, as well as,

no implementation for real-time applications.

Contiki provide dynamic memory management and dynamic linking of programs. Against memory fragmentation, it is used a set of mechanisms: memory block allocation; managed memory allocation; as well as, the standard C memory allocator (malloc) [119]. Although, it is worth notice that Contiki does not provide any memory protection mechanism between different applications [43].

Communication support is available for both IPv4 and IPv6, contains a uIP implementation (a TCP/IP protocol stack for small 8 bit micro-controllers) supporting TCP, UDP, ICMP and IP protocols. Contiki also supports single hop unicast, single hop broadcast, and multi-hop communication. But it does not support multicast. Contiki provides an implementation of IPv6 Routing Protocol for Low power and Lossy networks (RPL) [120], named ContikiRPL [121]. ContikiRPL operates over low power wireless links and lossy power line links.

Regarding the way to access data, the Contiki OS, uses a file system called Coffee [122]. Coffee is a flash-based file system, that uses a small and constant footprint for each file, based on the concept of micro logs to handle file modifications without using spanning log structure. The Contiki also gives support to other features, such as sensor network simulations through Cooja [123], security for communications using ContikiSec [124] and documentation can be found at http://www.contiki-os.org/, the Contiki official page.

2.3.4.3 MANTIS

MANTIS, the MultimodAl system for NeTworks of In-situ wireless Sensors [125], provides a lightweight, multithreaded and energy efficient operating system. With a footprint of less than 500 bytes RAM, 14KB flash, which includes kernel, scheduler and network stack. The MANTIS Operating System (MOS) main feature is that is portable across multiple platforms, and supports sensor nodes remote management through dynamic programming. MOS is written in C.

MOS follows a layered architecture. Supports pre-emptive multitasking due to the facts presented in [126] i.e.: *"A thread driven system can achieve the high performance of event based systems for concurrency intensive applications, with appropriate modification to the threading package"*. MOS maintains two logically distinct sections of RAM. One for global variables (allocated at compilation time) and another managed as a pile. Data concurrence conditions are avoided using binary multiplexers and semaphores.

MOS uses pre-emptive priority-based scheduling through the use of a UNIX-like scheduler, with multiple priority classes and round robin approach within each class. The energy efficiency is obtained by the scheduler by switching the microcontroller to sleep mode when the threads are in idle mode. MOS uses priority scheduling (that may achieve better results that the TinyOS or Contiki schedulers). However, it is still required the use of real-time schedulers, such as Rate Monotonic and Earliest Deadline First (EDF).

For this reason it is stated that, also MANTIS does not provide support for real-time applications.

MANTIS allows dynamic memory management, but it does not recommend it, because its use incurs in lots of overhead. Also, MANTIS does not provide any mechanism for memory protection. Resource sharing is achieved through the help of semaphores.

Regarding communication protocols, MOS provides an implementation through layers. Network stack at a higher level (at the same level that users applications), and MAC and PHY at a lower level layer (at kernel level) call COMM layer. MOS does not provide support for multicast applications or group management protocols. Although, it provides the capability to implement custom routing and transport layer protocols.

MANTIS OS also provides support for wireless sensor network simulation through Avrora [127], an implementation of a Unix-like shell that runs on sensor nodes. MANTIS documentation can be found at https://sourceforge.net/projects/mantisos.

2.3.4.4 WSN OSs Summary and Comparative Analysis

Previous sections presented the three most popular Operating Systems (OS) for Wireless Sensor Networks (WSN) [42, 43]. These OSs are TinyOS [108], Contiki [116] and MANTIS [125]. Also, a work in [128], presents a study regarding the number of publications/articles related to each OS (see Figure 2.8). The main scientific and engineering online databases analysed were IEEE Xplore, ACM Digital Library and Science Direct. It is possible to observe that TinyOS is much more used by the scientific community in comparison with the other OSs.

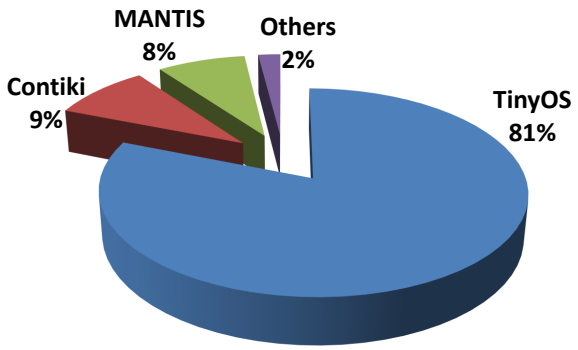


Figure 2.8: Articles Percentage related to each Operating System.

Next is presented a summary of some analysed features (see Table 2.3), and is performed a comparative analyse of the mentioned OSs. Furthermore, in Table 2.4, is presented a short list of hardware platforms and the correspondent OSs which supports them.

In Table 2.3 is shown a column, 'Year', in which first is specified the date of the first known version and then when was the last known OS change. It is possible that MANTIS OS is already forgotten, and therefore not a viable solution, since there may no longer exist development support.

Table 2.3: Summary of WSN OSs Features.

OS:	TinyOS	Contiki	MANTIS
Year:	1999 (vs. 0.43); 2012 (vs. 2.1.2)	2002; 2013 (vs. 2.7)	2003; 2007 (vs. 1.0 beta)
Architecture:	Monolithic	Modular	Layered
Execution Model:	Event-based; Multithread (TOSThread add in vs. 2.1)	Event-driven Kernel; Multithread (Protothreads)	Multithread
Program Language:	nesC	C	C
Footprint:	Bytes Order	KB Order	KB Order
Scheduling:	FIFO and EDF	Events run to completion; Interrupt handlers to post new events	Priority-based scheduling
Memory Management and Protection:	Memory Protection; Static Memory Management	Dynamic linking and memory management; No memory protection	Dynamic memory management but it not recommend; No memory protection
Resource Sharing:	Virtualisation and Completion events	Serialized access	Binary mutexes and semaphores
Real-time Applications:	No	No	No
Communication Support:	6lowpan; IPv6; DYMO; DIP; IEEE 802.15.4	IPv4, IPv6, uIP, RPL, ContikiRPL	Layered support; Networking at high level layer, MAC and PHY in a lower layer
Communication Security:	TinySec	ContikiSec	Not Available
File System:	Single level file system	Coffee file system	Not Available
Simulation:	TOSSIM/ Power-TOSSIM[129, 130], Avrora [127]	Cooja [123]	Avrora [127]

The OSs uses two types of execution models. The early ones (TinyOS and Contiki) apply an event-based execution and MANTIS apply a thread-based execution model. Nevertheless, TinyOS and Contiki have already added multithread support to their systems. The main difference identified in Table 2.3, is the footprint size of each one of the considered OSs, where the TinyOS stands out by its small size (400 bytes).

TinyOS has memory protection, while the others do not. On the other hand, Contiki

have the capability to perform dynamic memory management (MANTIS also present support, but do not recommend its use). Resource sharing is a functionality presented by all, in one way or another.

TinyOS and Contiki provide abstractions to access data in the shape of file systems. They use a single level file and Coffee file respectively. The advantage in using this kind of approach is that users are able to manage entities, identified through file names in order to access data. It facilitates programming and prevent possible errors, since the user do not need to be aware about low-level details (e.g. the number of pages to be read/write). Simulation support is available for all presented OSs.

Table 2.4 displays a list of hardware platforms supported by each one of the presented operating systems. The author would like to point out that the content of this table was produced based on several articles from the scientific community, from the OSs home pages and from hardware platforms manufactures. This is valid up to the date of the document elaboration.

Table 2.4: List of supported Hardware platforms.

	TinyOS:	Contiki:	MANTIS:	References:
Mica	✓			[43, 109, 131]
Mica2	✓		✓	[43, 131–133]
Mica2Dot	✓		✓	[131, 134]
MicaZ	✓	✓	✓	[43, 119, 131, 133, 135, 136]
TelosA	✓		✓	[43, 131, 133, 135]
TelosB	✓	✓	✓	[131, 133]
Tmote Sky	✓	✓	✓	[43, 131]
AVR series MCU		✓		[43, 119, 137]
BTnode3	✓			[131]
Rene	✓			[131]
Eyes	✓			[131]
Imote	✓			[131, 135]
Imote2	✓			[131, 133]
Cricket	✓			[131]
TinyNode 584	✓			[131, 133]
Mulle	✓			[131, 133]
SenseNode	✓			[131]
Iris	✓	✓		[119, 131, 133]
MANTIS nymph			✓	[131]
JCreate		✓		[131]
Atmel Raven		✓		[131]
MSB		✓		[131]
ESB		✓		[131]

2.3.5 Security

Security is a vital aspect in any situation and in IoT is not different. IoT security and privacy are critical themes, that must be addressed with a careful attention so IoT services can be safe and reliable. Methods must be applied taking into consideration the Resource-Constrained nature of IoT Systems, since cryptography approaches normally need high level of computation skills and this leads to a higher consumption of energy.

Many publications [138–141] highlight three security dimensions as more important:

- **Authentication and Confidentiality:** systems and data is secure and can only be access by legal, authorised users. A user can be human but also machines and services;
- **Integrity:** with data exchange being a top aspect in IoT, is crucial to ensure that data, information is not tampered. Ensure data accuracy;
- **Availability:** systems, data should always be available whenever it is needed.

The increasing number of available IoT Systems connected to the network, providing, working also to provide a large number of services has been putting an extra pressure to service providers with the increasing risk involving users. To address this, and the weaknesses of traditional security policy (a high cost in computation and energy), approaches have been focus on an emerging and promising new technique to address wireless communication security — Physical-Layer Security (PLS) technologies [142].

An overview regarding the other levels of standardisation, presented in the previous sections, gives a small idea how security is been addressed and which approaches are been applied for the different IoT levels. Organisations such as OASIS and ISA are focus on developing security solutions. MOM protocols use different approaches such as SASL, SSL, TLS, DTLS, IPsec, application dependent, etc. Two of the three addressed OSs presented communication security, the TinyOS with TinySec [114] a link layer security, and Contiki with ContikiSec [124] a network layer.

2.4 Topic Discussion

Internet-of-Things (IoT) is considered to be a highly heterogeneous environment, composed by a vast number of "things" (devices, sensors, smart objects, etc.), sometimes called the Internet of Objects. IoT is referred as an Internet evolution, a new stage of the Internet [143]. It is a network of objects, things, systems, applications and people. The Micro-Electro Mechanical Systems MEMS had a great influence in such evolution, contributing to smaller and cheaper sensor nodes with capability to communicate and share information among each other [3, 11].

This section started by presenting the importance which Resource-Constrained Systems (RCS) have on IoT. They have become increasingly important to society, with strong

capabilities to give people a more truly environmental awareness and interaction with the surrounding world. However, as the name implies, these devices (IoT Systems) present some constraints regarding available resources, such as low power, small processing and storage capabilities.

It was also identified that the main constraints for IoT development are: scalability; standardisation, specifically to security and privacy; and energy consumption. The author would like to enhance the energy efficiency and the diversity of IoT Systems available. Although, usually challenges related with cost and design robustness are also considered important aspects.

Identified as one of the IoT main issues, standardisation was addressed covering different areas like communication and data protocols, platforms, embedded OSs, security aspects and organisations that are focus on standards development and its general use.

With the standards small review (a complete standards survey is a very long theme, time expensive and complex), is possible to state that the number of solutions is wide independently of the aspect we intend to address. Section 2.3.2.1 presented a narrow set of possible wireless communication protocols (e.g.: NFC, Wi-Fi, LoRa, GPS, etc.). Section 2.3.2.2 addressed data protocols focusing on MOM protocols (e.g: AMQP, MQTT, CoAP). Section 2.3.4 present only three OSs used to build applications for RCS. These are relevant, sometimes contradictory features when thinking in the design of one IoT System.

All pushing in one direction, a vast number of possibilities that put together create an infinity number of possible solutions. The number of possibilities of each theme goes far beyond what it was presented in this section. Powered with the diversity, heterogeneous devices' nature, is clear that to complete define an IoT System is needed formalisms capable of digitally describe a hardware platform, application code (that will impose for example the definition of communication and data protocols) and if available information regarding energy consumption is very welcome.

From such descriptions will be possible to build, define tools, methods that could assist on decisions regarding the more suitable IoT System for a specific purpose, or as input for simulation tools, to better predict devices lifetime in real deployments as well to serve as diagnostic tool for a more energy-efficient applications.

Chapter 3

Model-Driven Approaches

Over the past six decades, researchers and developers have been trying to create high level software abstractions for applications development and for systems design-phase. These abstractions focus on language as in platform technologies [144]. Programming language evolved from machine code, passing through assembly and Fortran, to today's more common programming language — object oriented. The same happened with platform technologies, developers left behind the complexity of having to program directly the hardware. The evolution of Operating Systems (OS) is a good example.

From the efforts made in past, to increase abstraction levels in software development, is important to emphasize in 1980s the Computer-Aided Software Engineering (CASE) and 2000s the Model Driven Engineering (MDE). The next sections will focus on these two topics.

3.1 A First Attempt: Computer-Aided Software Engineering

In early computers, programmers create their own systems by literally coding instructions in 1s and 0s. Then an important software innovation arise — the assembly language, followed by other programming languages with different levels of abstraction. However, it was highly inefficient, expensive and timeless. Other types of solutions were needed.

Computer-Aided Software Engineering (CASE) focuses on developing software methods and tools, to automate different activities in systems development life cycle phases. It is a computer-assisted method to organise, control software development on large and complex projects, which can also involve many people and project parts. For example, developers can express their ideas at the design phase, through graphical programming

representations, such as dataflow diagrams, state machines or structure diagrams. Programmers then could use automated code generators to convert the graphical representations into code [144, 145].

One of CASE goals is to enable a more accurate and better analysis of software programs. But it can also identify memory leaks and corruption associated to programming languages. It is able to perform identification of repeated implementation parts, from the graphical representations, reducing code effort and debugging [144].

In sum, CASE enables the implementation of automated development tools, to increase organisations productivity, decrease costs, improve project controls, and improve products quality.

3.1.1 CASE Risk Factors

CASE attracted considerable attention in research community and trade literature. Although, it was not widely adopted, since graphical representation languages were mapped poorly to languages and main platforms. The lack of Quality-of-Service (QoS) properties (e.g. fault tolerance and security), lead to large amounts of code to compensate. Making it hard to debug and improve tools, with CASE tools [144].

CASE presents common risks associated to the use of automated technologies to develop/improve systems. First an inadequate standardisation, CASE tools from different vendors use different code structures and data classifications, which make it difficult to systems interoperate. File formats can be converted, but usually it is not economically viable. Second point, is the unrealistic expectations. Implementing CASE strategies usually involves an early investment, with a long-term payback. Another point is quick implementation. Organisations should not use CASE tools for the first time on critical projects or projects with short deadlines. Implementation of CASE tools involves significant change to the traditional development environments [146].

3.2 Model-Driven Engineering

Nowadays, IT managers, entrepreneurs and software developers use models for almost anything. The constant increase of systems complexity, shorter development cycles and high quality expectations, has push forward, the use of model driven techniques on modelling/design critical stages of systems development. With the use of model driven approaches, models have become primary artefacts in software development [147].

This trend, Model Driven, has been followed by academia and industry regarding not only automatic software development, but also enterprises integration and interoperability. It emerged from efforts such as Model-Driven Architecture (MDA) [148], Model Driven Development (MDD) [149] and Model Driven Engineering (MDE) [144]. Figure 3.1 overviews the relations and evolution of these three initiatives. The presented

dates refers to the first journal publication, as mentioned in [150], and not to the first appearance.

MDE is defined in [144] as: *"Model-driven engineering technologies offer a promising approach to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively"*. MDE, in some cases, called Model-Driven Development (MDD), is an approach to tackle systems complexity through simplification and formalisation techniques during system life cycle (i.e. from design to deployment, passing by construction, operation, modification, etc.) [23].

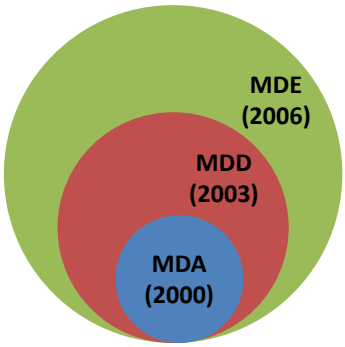


Figure 3.1: MDA, MDD and MDE initiatives.

3.2.1 Models and Meta-Models

With the promising software engineering approach, Model Driven Engineering (MDE), comes the buzz words — Model and Meta-Model. A basic principle that was very helpful in engineering along the years, and still is, states that *"everything is an object"*. This generalisation has pushed forward to simplicity and systems integration. In MDE, there is a similar principle, which states that *"everything is a model"* (i.e. entities, software, services, platforms, etc.) [151]. Next is presented some notions and definitions regarding what is a Model, what is a Meta-Model and their relationship.

A model in software engineering is traditionally referred as an artefact, represented/described using a modelling language. Usually, model descriptions are graph based, using tools such as the UML [152]. Models in software engineering are divided in two types: ***descriptive***, which are used to gather knowledge, e.g. requirements, analysis, etc.; and ***prescriptive***, models used as blueprints for systems design, implementations, etc. [153].

According to the work presented in [152], a model contains information of something (content, meaning), created by someone (source), for somebody (target), with some purpose (usage, context). Among many other descriptions concerning what is a model, Webster's new encyclopaedic dictionary [154], states that *"A model is a description of something"*. Within MDD scope, models are linguistic by nature, since they are expressed by some language. The author will adopt this characterization from now on.

Another definition of a Model is that it captures a portion of reality that is being observed and interpreted. A Model uses relationships, abstract elements and properties from a variety of languages, concepts and formalism levels to represent real-world entities and its relationships and properties [155].

A model can also be seen as a set of statements/specifications regarding a System Under Study (SUS) or a class of the SUS. A model interpretation is the recognition in which ways it could be mapped to SUS or to partial elements of SUS [156].

What is a Meta-Model? Examining the "Meta-Model" word, the prefix "meta" means literally "after" in Greek; and is used whenever an operation is applied twice (in this case — to Model). Several Meta-Model definitions can be found, such as *"A Meta-Model is a model of models"* [148] and *"A Model is an instance of a Meta-Model"* [157].

Meta-Model is a specification model for a modelling language, where SUS classes are expressed. Those classes are the SUS model itself. In sum, a Meta-Model specifies, through the modelling language, what statements can be made within the Model, to describe a system. As stated, a Meta-Model is by its' own a Model. Consequently, must be also written in a coherent modelling language — Meta Language. This Meta Language is responsible for the specifications, in which a Meta-Model is expressed. Accordingly, a Meta-Model describing the Meta-modelling language must also exist, in the same way as in the Meta-Model/Model relation [158].

Established the notion of what is a model and what is a meta-model, now it is important to consider the rules which they have to obey. Models must be written in a well-defined modelling language, and its relationships, classes, attributes that form a SUS should support/follow the unification principle, as described in Figure 3.2. The SUS should be described in a coherent way, syntactic as well as semantically.

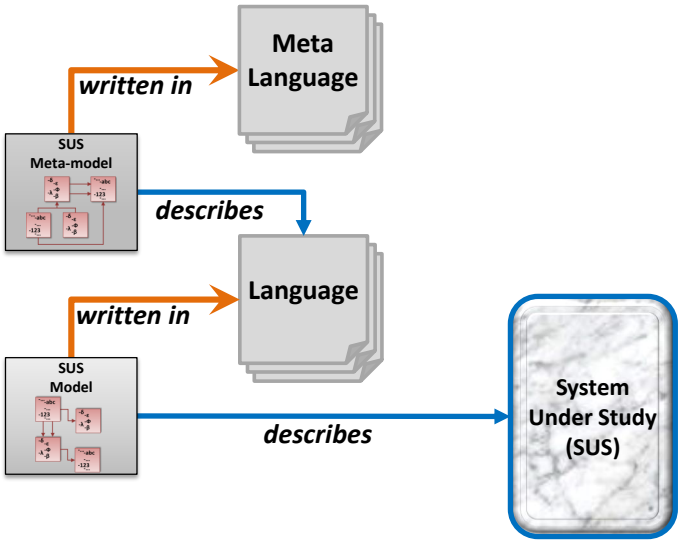


Figure 3.2: Model and Meta-Models Relationship.

As stated before, modelling languages are described by Meta-Models. A Meta-Model it is nothing more than a model specifying constructs and relationships used in a given

modelling language. Meta-Models define solid statements about what can be expressed in a valid model. Another imperative condition is that a model must be in conformance with its meta-model. It must not violate any statement and construct rules inherent or deductible from the meta-model [158].

3.2.2 Model-Driven Architecture

Several approaches have been made available which follows MDD/MDE principles. Among these can be found: Agile Model Driven Development [159], Domain-oriented Programming [160], Microsoft's Software Factories [161] and Model-Driven Architecture [162]. MDA is considered to be the more prevailing approach to this moment [23].

MDA, an OMG initiative, was proposed in November 2000 [163] to address interoperability issues, based on the MDD principles. Object Management Group (OMG) is an international, open membership, not-for-profit technology standards consortium, founded in 1989.

MDA is one example of the broader MDE vision, covering many popular, current research trends related to generative and transformational techniques in software engineering, system engineering, or data engineering [164, 165]. It uses a well-known and long established idea of keeping apart the systems operation specification, from the details of how systems use their own platform capabilities.

MDA provides an approach for, as well as, it enables tools to provide:

- system specification independently of the platform that supports it;
- platforms specification;
- choosing a particular platform for the system;
- and transforming the system specification into one for a particular platform.

The three primary goals of MDA are portability, interoperability and reusability through the separation of architectural systems parts [148].

3.2.3 MDA Models and Viewpoints

The Model-Driven Architecture (MDA) considers three viewpoints: the computation independent viewpoint; the platform independent viewpoint; and the platform specific viewpoint. For each one of the viewpoints, MDA considers a different model.

MDA specifies a standard-based architecture for models, providing a set of guidelines to structure specifications, organised around three different abstraction levels (viewpoints) [148, 166, 167]: Computation Independent Model (CIM); Platform Independent Model (PIM); and Platform Specific Model (PSM). The three abstraction levels organisation is depicted in Figure 3.3.

The computation independent viewpoint focuses on the environment in which the system will operate, and on system requirements. The **Computation Independent Model (CIM)** is a system view from the computation independent viewpoint. The structure and processing system details are hidden or even not determined at this level. It is also called a domain model or business model, since it is meant for the domain practitioners and it is based on the specific target domain vocabulary. In this sense, it is useful, not only as an aid to understand a problem, but also it plays an important role in bridging the gap between domain experts and development experts, that will build systems and/or services [148].

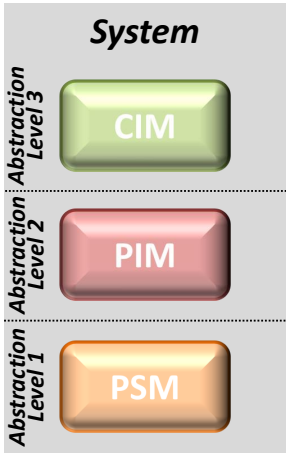


Figure 3.3: MDA's Abstraction Layers (viewpoints).

The primary user of CIM (a domain expert), is not familiarize with models or artefacts, that further on will accomplish the functionality for which the requirements are being articulated by the practitioner at CIM level. Therefore, typically such a model is independent of how the system is implemented.

In a MDA system specification, the CIM requirements should be traceable to PIM and PSM level constructs, which will implement them, and vice versa. A system CIM might be a view of one or several viewpoints, each one including one or several models for those viewpoints. Some can provide more detail than others, or focus on particular concerns of a viewpoint.

Platform independent viewpoint focuses on operation details, while it hides specific details of any particular platform, so its use can be suitable to several different platforms. The **Platform Independent Model (PIM)** is a view of a system, from the platform independent viewpoint. It presents a certain degree of platform independence, so it can be applied to different platforms of similar type.

The PIM describes part of a complete specification, the part that does not change from one implementation platform to another implementation platform. It is used to build formal specifications for structure and functionality of a system, abstracting technical details. PIM uses a general purpose modelling language or a specific language to the area in which the system will be applied.

A platform specific viewpoint combines the platform independent viewpoint with details that specify how the system uses a particular type of platform. The **Platform Specific Model (PSM)** is a system view from the platform specific viewpoint.

A PSM can provide more or less detail depending on its purpose. It can add to PIM, technical details and implementation constructs that are available in a specific implementation platform, including middleware, operating systems and programming languages (e.g. Java, C, C++, VHDL, nesC, etc.). Or, it can be used as a refinement to the PIM, to further on produce a PSM that can be implemented directly. The PSM is a model of the same system specified by the PIM, but it also specifies how that system makes use of the chosen platform.

3.2.4 MDA and Model Transformations

Model-Driven Architecture (MDA) is a software design approach, built to shorten software development life cycle, as well as, enhance systems readability for its stakeholders. It presents a standard-based architecture for models, providing a set of guidelines to structure specifications, which are organized around three different abstraction levels [148, 166, 168].

To maintain the relationship between the different abstraction levels and different systems, two kinds of transformations were defined: **horizontal** and **vertical transformations** (see Figure 3.4). It is important to notice that a system can be or include anything: a program, a single computer system, some combination of parts of different systems, etc.

Model transformation, i.e., the process of converting one model to another model, is an important activity in MDE/MDA. Transformations are realised through mappings, from a source model to a target model, using a specification description, transformation rules, or other type of information. Each transformation type (horizontal or vertical) can be accomplished using different approaches: marking, meta-model transformation, model transformation, pattern application or model merging.

3.2.5 Horizontal and Vertical Transformations

Horizontal transformations (see Figure 3.4) do not affect the abstraction level of the model. When performing a horizontal model transformation (e.g. refactoring of individual models, language translation, or even joining different models), an explicit or an implicit mapping of the "meta-model" has to be performed. This type of transformations is normally a static process, but due to the constant knowledge change, services, models and ontologies that regulate systems are not static. This constant change is mainly caused by the dynamics of global market.

As explained before, MDA specifies a standard-based architecture for models, which is organised around three different abstraction levels. Since the details of each modelling level are different, to perform such transformations (vertical transformations) is necessary

to define a consistent set of rules that provide extra meaning to components from one abstraction level to another. These components usually represent real objects, interactions, behaviours, services or systems which are represented thanks to a countless number of modelling languages, with completely different types (e.g., graphical, object-oriented, etc.) that are able to establish rules for model transformation.

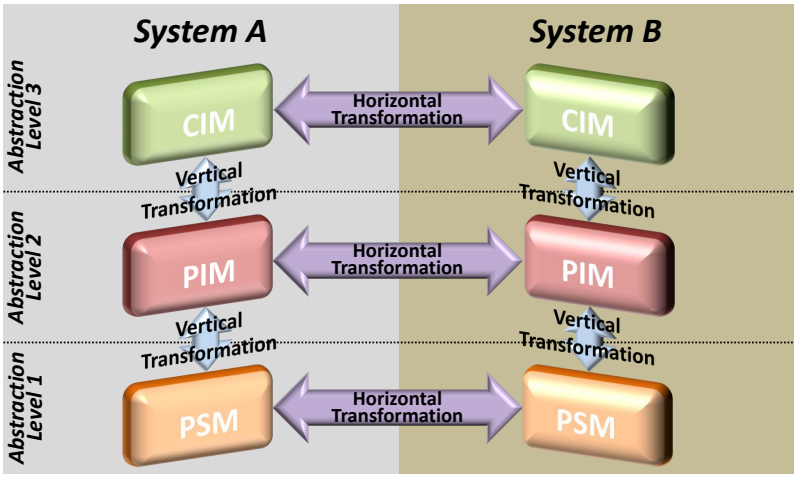


Figure 3.4: MDA's Abstraction Layers and Transformations.

Vertical Transformations (see Figure 3.4) can be achieved through the use of MDA tools to translate from one abstraction level to another (i.e. refine or abstract a model; they affect the abstraction level of the model specification). Most of the MDA tools provide a mechanism to perform model annotations at the different abstraction levels. Additionally, they provide means to customize transformation rules according to user's needs, as well as, predefined PSMs, along with their respective annotation stereotypes for the most commonly used programming languages (e.g. Java, C++, etc.). The amount of generated code depends on both the code generator and also on the level of detail represented in the PSMs (i.e. how well the PSM captures the details of the physical platform).

3.3 Modelling and Transformation Languages

The Model-Driven Development (MDD) can be seen as an approach to create applications tackling the complexity through simplification and formalisation techniques during the design phase. Model-Driven Architecture (MDA) is an OMG initiative to address tools and systems interoperability based on MDD principles, enabling also the transformation from systems specification (e.g.: design models) to a particular, specific platform (e.g.: Java).

Next section will address two important topics regarding model-driven standard technologies, the modelling and transformation languages, which enable the implementation of MDD approach.

3.3.1 Modelling Languages

The Object Management Group (OMG), organisation responsible for Model-Driven Architecture (MDA), has adopted some technologies, which together enables the model-driven approach. These technologies include the Unified Modelling Language™ (UML), the Meta-Object Facility (MOF)™, Extensible Markup Language™ (XML), the XML Metadata Interchange (XMI), and Ecore the Eclipse Modeling Framework (EMF) Meta-Model. Next section will present an overview regarding these modelling languages.

3.3.1.1 Unified Modelling Language (UML)

The Unified Modelling Language (UML) is a standard modelling language that allows visualising, specifying and documenting software systems. The main purpose of UML is to provide system architects, software engineers, and software developers with tools to analyse, design, and implement software-based systems as well as modelling business and similar processes [157].

However to accomplish these objectives UML has to fulfil some requirements to enable meaningful exchange of models information between systems, such as:

- "A formal definition of a common MOF-based meta-model that specifies the abstract syntax of the UML. The abstract syntax defines the set of UML modelling concepts, their attributes and their relationships, as well as, the rules for combining these concepts to construct partial or complete UML models."
- "A detailed explanation of the semantics of each UML modelling concept. The semantics define, in a technology-independent manner, how the UML concepts are to be realised by computers."
- "A specification of the human-readable notation elements for representing the individual UML modelling concepts, as well as, rules for combining them into a variety of different diagram types corresponding to different aspects of modelled systems."

3.3.1.2 Meta-Object Facility (MOF)

Meta-Object Facility (MOF) is based on a simplification of the UML 2's class modelling capabilities which provides the basis for a meta-model definition for the MDA languages, with core capabilities for model management, regardless of applied meta-model. There are now two compliance parts: Essential MOF (EMOF) and Complete MOF (CMOF) [169].

MOF Core defined in ISO/IEC 19508, is the technological foundation to describe meta-models, providing also models repository which can be used to specify and manipulate models in all the MDA phases. Through the MOF platform-independent meta-data management framework, with a set of meta-data services associated to it, MOF enables the development and interoperability among heterogeneous models and model-driven systems.

3.3.1.3 eXtensible Markup Language (XML)

The eXtensible Markup Language (XML) is a simple, very flexible text format designed by World Wide Web Consortium (W3C) to store information that can be read by machines, as well as, by humans. XML distinguish document's contents, structures and presentations, by using XML tags, each one with its own meaning, to represent, identify blocks of data.

XML Meta-Model is composed by five classes: ***Node*** (core and abstract class), ***Element***, ***Attribute***, ***Text*** and ***Root***. The classes ***Element***, ***Attribute*** and ***Text*** inherit from class ***Node***, while ***Root*** inherits from class ***Element***. ***Node*** class main attributes are "*name*" and "*value*". Class ***Element*** contains a reference, "*children*" to enable inclusion of successive classes of type ***Node***. It also references a package, named ***PrimitiveTypes***, with three types of primitive data: boolean, integer and string.

Application examples of XML utilisation go from saving and displaying a contact list, to integrate multiple independent systems, through interfaces defined in XML. There are currently a number of initiatives to replace proprietary formalization methods by XML-based methods due to simplicity inherent to XML use.

3.3.1.4 XML Metadata Interchange (XMI)

The XML Metadata Interchange (XMI) main purpose is to simplify the meta-data interchange between development lifecycle tools in distributed heterogeneous environments [170]. Among these tools is possible to find tools based on UML, ISO/IEC 19505, and meta-data repositories/frameworks based on the MOF, ISO/IEC 19508.

The XMI integrates three key industry standards: the eXtensible Markup Language (XML), a W3C standard, the UML and MOF from Object Management Group (OMG). XMI is widely used as the interchange format for XML files. It defines a set of aspects to describe objects in XML, such as:

- The representation of objects in terms of XML elements and attributes;
- The standard mechanisms to link objects within the same file or across files;
- The validation of XMI documents using XML Schemas;
- Object identity, which allows objects to be referenced from other objects in terms of IDs and UUIDs.

3.3.1.5 Ecore: Eclipse Modeling Framework (EMF) Meta-Model

Eclipse Modeling Framework (EMF) is a very popular modelling tool that is part of the Eclipse modelling project, Eclipse Modeling Project (EMP). The Ecore modelling language is the EMF Meta-Model [171]. EMF also provides runtime support with default XMI serialization, and a very efficient reflective API to manipulate EMF objects in a generic way.

Ecore's flexibility allows to model complete hierarchies of Domain-Specific Languages (DSL), such as models "sequence", each being the meta-model of the next. Regarding code generation, EMF provides tools for code generation from Ecore models and their instances.

3.3.2 Transformation Languages

A model-based architecture to accomplish interoperability between models, a mapping from the source to the target model must be defined. However, to implement such conversion a type of methodology has to be applied. That methodology is based on transformation languages.

A specific standard language for model transformation has been defined by OMG called Query/Views/Transformations (QVT). Although, is Atlas Transformation Language (ATL) that is widely used, due to a large user database and its good integration with JAVA. ATL was developed by OBEO and INRIA, inspired in MOF QVT. Both transformation languages will be addressed in the next sections.

3.3.2.1 Query/Views/Transformation (QVT)

The mapping process between models is expected to correlate one or more input models (source models) with one output model (target model). However one source model can be mapped to several target models. Mapping languages are languages that execute the transformations between models, i.e. the models mapping.

Model transformation is an important activity in MDE as it has been recognised by the OMG. OMG issued the MOF Query/Views/Transformation (QVT) Request for Proposals (RFP) to seek an answer compatible with its MDA standard suite. Many contributions for the QVT (RFP) were submitted, like ATL and QVT itself. Comparisons of applicability and interoperability between several transformation languages are widely available [172], which helps narrowing the choice to a few transformation languages for a known given type of transformation.

The Meta-Object Facility (MOF) QVT language defines three related transformation languages: Relations, Operational Mappings, and Core. The QVT specification depends on two OMG specifications: the MOF 2.0 and the Object Constraint Language (OCL) 2.0 specifications. The QVT specification is divided in two parts, a declarative part formed by two layers, a user-friendly Relations meta-model and a Core meta-model, and the imperative part formed by the Operational Mappings Language and the Black Box implementations [173], as depicted in Figure 3.5.

The user-friendly Relations layer consists in a declarative specification of the relationships between MOF models. The Relations language is capable of performing complex object pattern matching and is able to create a trace of what happens during a transformation. The Core is a small model/language that only supports pattern matching, evaluating the conditions over a set of variables against a set of models. It deals with the source and

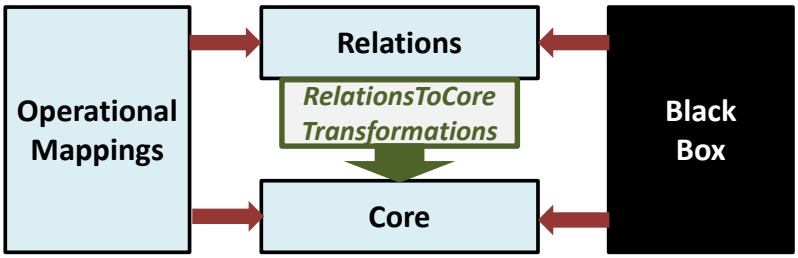


Figure 3.5: Relations between QVT Meta-Models (based on [173]).

targets elements as well as the trace models in a symmetrically way. However, the Core trace models must be explicitly defined, since they are not retrieved from the transformation description as in the Relations case. Concerning the imperative Operational Mappings, it is considered to be a language which provides imperative implementations, fulfilling the same trace model as the Relations language. Operational Mappings can be used to implement one or more Relations. On the other hand, a Black Box implementations do not have an implicit relationship to the Relations layer, therefore each one of these implementations must explicitly implement a Relations to keep trace between the model elements.

3.3.2.2 ATLAS Transformation Language (ATL)

Atlas Transformation Language (ATL) is a transformation language and a tool kit developed and maintained by OBEO and AtlanMod (INRIA). In the field of MDE, ATL provides ways to produce a set of target models from a set of source models, with capabilities to perform syntactic or semantic translations.

The ATL is a hybrid language capable of performing declarative and imperative transformations and it is specified as a meta-model, as well as, a textual concrete syntax. An ATL transformation is composed by rules that define how the elements from the source model are matched and navigated, and also how the target models elements are initialised.

The Figure 3.6 shows how a transformation is defined following the OMG MOF/QVT compatible with the MDA. A given transformation operation is then represented as follows:

$$ModelB \leftarrow f(MetaModelA, MetaModelB, ModelT, ModelA) \tag{3.1}$$

Equation 3.1, means that a target model (Model B) based on a target meta-model (Meta-model B), is obtained from a source model (Model A) based on a specification model (Meta-model A), by applying a transformation (Model T) based on the standard transformation language (i.e.: ATL).

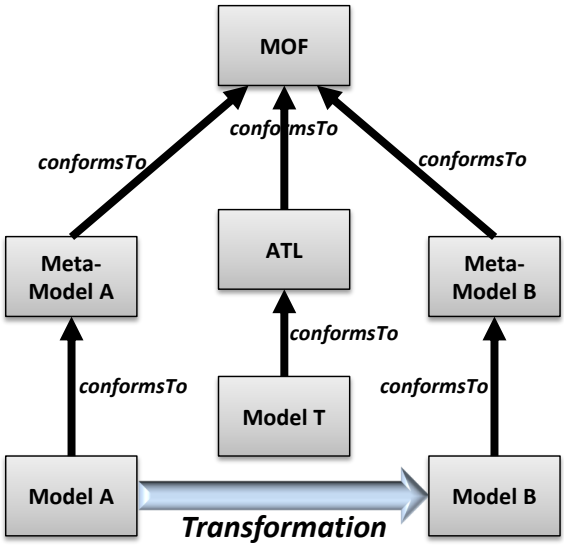


Figure 3.6: A Generic ATL Transformation.

The ATL defines its abstract syntax by the way of a meta-model (see Figure 3.6) taking inspiration from the Object Constraint Language (OCL) 2.0, which may be considered here as an assertion and as a navigation language at the same time. This means that every ATL transformation is in fact a model, with all the properties that are implied by this [174].

3.4 Topic Discussion

Model-Driven Engineering (MDE) methodology is addressed in this literature review, due to its importance and significant impact on automatic tools development. As stated before, MDE has been used by the academia and industry for software development [147]. Its capabilities to formalise and describe "anything" using models enables IT managers, entrepreneurs and software developers to tackle systems complexity by simplifying design and development processes [23].

Looking to the principle of "*everything is a model*" [151], an IoT System is no different. Model-driven techniques can then be used to formally describe hardware, software or energy aspects of an IoT System. Assisting in this way, on tools development based on such descriptions. For example, energy consumption-simulation, development of IoT applications by non-experts, multi-criteria assessment (of features such as cost, programming language, hardware characteristics), etc.

The Model-Driven Architecture (MDA) was initially design for software development, although it has been used in other areas like Business Process Modelling and systems/enterprises interoperability. MDA presents three abstractions layers, Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM), to cover a system development process. Although, MDA does not specify

transformations from CIM to PIM level since it uses an OMG standard, the Meta-Object Framework (MOF) which provides basis for a meta-model definition.

MDA presents two types of transformations, horizontal and vertical, in which a variety of possible transformation categories can be performed, such as, Marking, Meta-Model transformation, Model transformation and Model Merging. These transformations assist on the development process (more automatic) of new system functionalities, from the design phase to the actual implementation. It can also be used to perform system/enterprise interoperability. Matching processes phases or data descriptions from one company/system to another.

Chapter 4

Decision-Making Methodologies

Decisions are something that all people around the world have to face daily. Questions, either simple or complex, are placed constantly and decisions have to be taken. Decisions can be made by common sense, by lived experience or intuition from those who decide or advise. Although, when the number of criteria to be analysed increases, more difficult, it is to make a correct and accurate decision for human beings. Consequently, it is essential to organise information in a formal way, using available methods and techniques, to choose a best or more proper alternative to the question in hands.

Multi-Criteria Decision-Making (MCDM) is one of the most widely used decision methodologies in sciences, business, governmental and engineering worlds. MCDM methods can help to improve the quality of decisions, by making the decision-making process more explicit, rational, and efficient [24, 25, 175, 176].

MCDM is an approach comprehensive of techniques which provides alternatives ranking methods, methods to classify criteria of a certain problem and assist on decision phase. MCDM can tear down complex problems into small, manageable problems, in such way that small sets of data can be analysed from which is obtain reasonable judgements. Reassembling all pieces, it is formed a coherent overall system view that will aid decision makers on their decisions [13].

Two sub-classes derive from Multi-Criteria Decision Making problems. Multi-Criteria evaluation problems focus on choosing a solution from a finite, explicitly known number of alternatives made available in the beginning of the process. The second, Multiple-Criteria design problems focus on cases where alternatives are not known, infinite or not countable when the process starts. This sub-class is associated to multiple objective mathematical programming problems [177].

Following sections will focus particularly on MCDM sub-class: Multi-Criteria evaluation problems, i.e., problems with a finite and known number of alternatives. To meet

with the thesis scope, more particularly, assist on the selection of a proper IoT System from a set of available alternatives/solutions.

4.1 Decision-Making Process

One of the main challenges in engineering is the ability to choose from a set of alternatives which are the most correct and consistent ones. Given a specific situation, make a right decision is probably one of the toughest challenges for science and technology.

From the moment that a problem is identified until a solution is chosen, that best fit the defined requirements/criteria, it should be carried out some steps that aim to assist in the decision process [13, 178]. Figure 4.1 depicts a simplified version of this process.

Looking to the simplified vision of decision-making process, the first step, *Identification of Possible Solutions and Assessment Criteria*, identifies a set of possible alternatives/solutions for a problem, and which will be the criteria used to choose the most appropriate solution. Next, in *Solutions Evaluation based on Criteria* step, it is assigned a weight to each criterion, and solutions are evaluated based on the criteria importance. In the last step, *Results Analysis and Suitable Solution Identification*, it is performed the results assessment, to identify one or more suitable solutions for the identified problem. The process is in a close-loop because the identified solution can hold an unwanted parameter, not identified early, or a new criterion must be included or even it may be necessary to change a criteria weight.

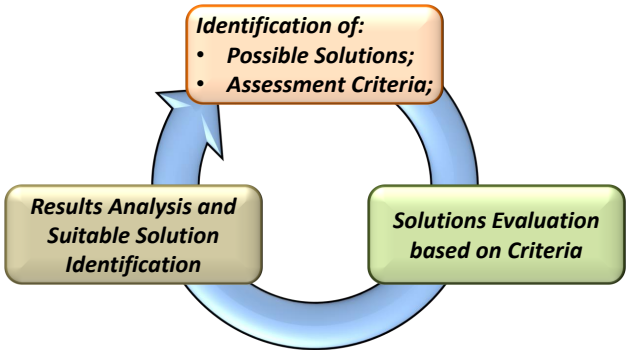


Figure 4.1: Simplified Vision of Decision-Making Process.

Identification of possible solutions is an obvious step in a decision making process. From terminology point of view, the term "possible solution" refers to an "option", "alternative", "possible action", to be applied or solve a certain problem. A possible solution must be formally characterised by a set of attributes which define the solution. Such attributes are then used to form a group of assessment criteria that allow a solution to be ranked.

To evaluate a solution, criteria are defined and must be specific, relevant and measurable, either in a quantitative or in a qualitative form. In the former case, it can be

considered simpler since it has a unit associated (e.g.: cost). The latter, in which is normally associated more vague concepts (e.g.: quality), it is necessary to use numerical indicators (more than one if necessary) or define categories (such as Very High, High, ..., or Very Low).

In multi-criteria problems there is no optimal solution, but a better, proper or suitable solution. Problems have often conflicting criteria and so it is impossible to choose a solution without controversy (except in a problem with non-conflicting criteria). The notion of suitable solution varies depending on decision maker. Each one has its' own interests and preferences. The identification of a solution must therefore include such opinion [178].

4.2 Methods for Multi-Criteria Decision-Making

Multi-Criteria Decision-Making (MCDM) is a process that can make a decision upon a set of available solutions, assessing which one is more suitable across diverse, contradicting, qualitative and/or quantitative criteria [177]. MCDM methods have been applied to engineering problems, providing useful insights to decision makers, making their decisions more qualified to overcome complex problems [179].

MCDM methods share a common ground, a starting point from which all methodologies are applied. This is that problems have a finite and known number of alternatives and solutions. From now on, ***Solutions*** are represented by set, S_{Set}, as in Equation 4.1 and the set of ***Features/Criteria***, called C_{Set}, is defined as in Equation 4.2.

$$S_{Set} = \{s_1, s_2, ..., s_n\}; n \in \mathbb{N} \tag{4.1}$$

$$C_{Set} = \{c_1, c_2, ..., c_m\}; m \in \mathbb{N} \tag{4.2}$$

Based in [179–181], as others could be point out, three of the most widely used decision making methods used in literature are the AHP (Analytic Hierarchy Process), Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) and Elimination and Choice Expressing the Reality (ELECTRE) (original name: Elimination Et Choix Traduisant la Realite).

4.2.1 Analytic Hierarchy Process (AHP)

Analytic Hierarchy Process (AHP) was proposed by Thomas L. Saaty [182], and it is aimed to solve problems with multiple and conflicting criteria. AHP is a powerful decision making methodology, which defines solutions ranking through a pair-wise comparison of multiple criteria. AHP original methodology has four steps, as shown in Figure 4.2.

The first step is to decompose a complex problem into a systematic hierarchy structure [183], as depicted in Figure 4.3. At the top is place the objective(s); next level contains all

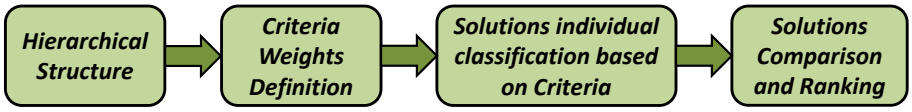


Figure 4.2: Analytic Hierarchy Process (AHP) original Methodology.

assessment criteria considered important to reach the goal; and finally, at the bottom are the solutions/alternatives.

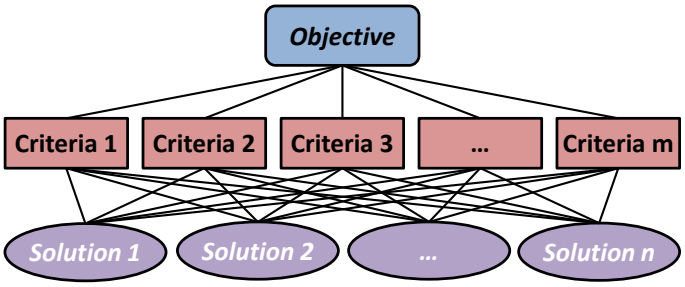


Figure 4.3: Example of a criteria and solutions hierarchy.

In the second step of AHP methodology, is applied a criteria pairwise comparison based on decision maker judgement, creating the Saaty 1-9 scale to assess criteria priority. Qualifying a criterion with 1 indicates equal importance and 9 extremely important (other levels are: equal importance, weak importance, extremely important, etc). With an m criteria and n solutions it is achieved a ***decision matrix***, ***DM***, of $n \times m$ size, in which v_{ij} element indicates the value for ith solution in respect to the jth criteria (see Equation 4.3). Where $i = \{1, 2, ..., n\}$ and $j = \{1, 2, ..., m\}$.

$$DM = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,m} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & v_{n,m} \end{bmatrix} = (v_{i,j}) \in \mathbb{R}^{n \times m} \tag{4.3}$$

The ***pairwise comparison matrix***, AHP_M, is an $m \times m$ size, in which the α_{kj} element indicates the value for kth criteria in respect to the jth criteria (see Equation 4.4). ***Normalized comparison matrix***, $NAHP_M$, is computed as presented in Equation 4.5.

$$AHP_M = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,m} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m,1} & \alpha_{m,2} & \cdots & \alpha_{m,m} \end{bmatrix} = (\alpha_{k,j}) \in \mathbb{R}^{m \times m} \tag{4.4}$$

$$NAHP_M = \frac{(\alpha_{k,j})}{\sum_{k=1}^{m} (\alpha_{k,j})} = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,m} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,m} \end{bmatrix}; k = 1, ..., m; j = 1, ..., m. \tag{4.5}$$

The contribution of each criteria j to the problem objective, i.e. the ***eigenvector*** — W, is given by Equation 4.6 [184]. It is also possible to compute the criteria pairwise comparison consistency, by applying Equations 4.7 and 4.8. Where λ_{max} is the maximum value from ***eigenvector***, W. ***CI*** is the ***Consistency Index*** and m the number of criteria.

$$W = \frac{\sum_{j=1}^{m}(A_{k,j})}{m} = \begin{bmatrix} w_1 & w_2 & \cdots & w_m \end{bmatrix}; k = 1, ..., m. \tag{4.6}$$

$$\lambda_{max} = \frac{1}{m}\sum_{j=1}^{m}\frac{(AHP_M\ W^T)_j}{W_j^T} \tag{4.7}$$

$$CI = \frac{\lambda_{max} - m}{m-1} \tag{4.8}$$

Knowing the ***Consistency Index***, ***CI***, is then possible to compute the ***Consistency Ratio***, ***CR***, which indicates the consistency of the decision maker subjective judgement. ***Consistency Ratio***, ***CR***, is given by Equation 4.9. If $CR > 0.10\,(10\%)$, exists serious inconsistencies, on the other hand, if $CR \leq 0.10\,(10\%)$, the consistency degree is completely satisfactory [185]. The ***Random Consistency Index***, ***RI***, as the name implies, is a random value. There are several studies, each one presenting its own random consistency index table. ***RI*** tables are built from exhaustive tests, where the number of criteria varies, and several pairwise comparisons, AHP_M, or judgement matrixes are applied. Table 4.1 presents values from two to ten criteria, retrieved from [186]. However, ***RI*** values can be computed for a number of criteria higher than 10.

$$CR = \frac{CI}{RI} \tag{4.9}$$

Table 4.1: Random Consistency Index values, RI, from 2 to 10 criteria.

n	2	3	4	5	6	7	8	9	10
RI	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

In step three of AHP methodology, solutions are classified individually based on criteria. Individual score is obtained through a simple matrix multiplication, i.e.: the decision matrix, DM, multiplied by the eigenvector transposed, W^T.

The Analytic Hierarchy Process (AHP) follows the prioritization theory, dealing with complex problems that need multi criteria consideration simultaneously. Although, AHP presents some disadvantages, such the pairwise comparison is based on decision makers' subjective judgement, and criterion weight has a direct impact on the final score.

4.2.2 PROMETHEE

Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) [187] is a MCDM method, with six different versions based on ranking. The first version, PROMETHEE I, consists on partial ranking; PROMETHEE II on complete ranking; PROMETHEE III performs ranking based on intervals and PROMETHEE IV is a method for the continuous case; PROMETHEE V method has integer linear programming and net flows; and finally the PROMETHEE VI method includes a representation of the human brain [184, 188].

The PROMETHEE methodology can be described in five steps, as Figure 4.4 shows. The decision maker chooses a preference function, defined in two actions independently (e.g. Boolean function — value 0 or 1), and then it is applied to compare solutions. The comparison results and criteria values are used to form a matrix, on which methods are then applied. For example, PROMETHEE I to create a partial solutions rank and PROMETHEE II to obtain a final solutions ranking [189].

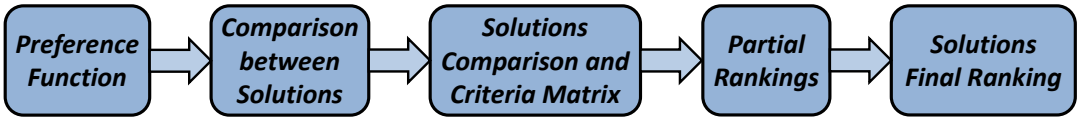


Figure 4.4: PROMETHEE Methodology.

Maintaining the focus on one of the thesis problems, i.e. evaluation of solutions with multi criteria, let's consider the problem defined in Equation 4.3. Decision matrix, DM, presents the criteria values for each individual solution, represented by v_{ij}, where ith solution in respect to the jth criteria.

Commonly decision problems present a multi-criteria nature, and its solution depends on the preferences made by each decision maker, consequently there is not a best absolute solution [184]. To represent such decision maker preferences are defined the following relations (see Equation 4.10):

For each (a, b), where $a, b \in S_{Set}$:

$$\begin{aligned} &\begin{cases} \forall j : v_{aj} \geq v_{bj} \\ \exists k : v_{ak} > v_{bk} \end{cases} \Leftrightarrow aPb, \\ &\forall j : v_{aj} = v_{bj} \Leftrightarrow aIb, \\ &\begin{cases} \exists s : v_{as} > v_{bs} \\ \exists r : v_{ar} < v_{br} \end{cases} \Leftrightarrow aRb \\ &k = 1, ..., m; j = 1, ..., m. \end{aligned} \tag{4.10}$$

Where P stands for $preference$, I for $indifference$, and R for $incomparability$.

The first equation branch (Equation 4.10), states that a solution a is better than another solution b if it is at least as good as the other in all criteria. But in one criteria k, solution

a is for sure better than b. Second equation branch indicates that solution a and b are equal in all criteria, therefore indifferent. The third and last branch, states that if there is a criteria s where solution a is better that b, and in criteria r, solution b is better than solution a, these solutions cannot be comparable, at least without additional information.

PROMETHEE methods use a preference modelling process, based on the information between criteria, and information within each criterion. Information between criteria is achieved by weight attribution to each criterion by the decision maker. Weights are non-negative numbers, and higher the weight, more important the criteria. It is common to use normalized weights. Information within criteria is obtain by a preference function, P_j, reflecting evaluation differences between two solutions for a specific criteria $\boldsymbol{j}$ [190]. Preference function result, P_j, as described in Equation 4.11, can be consider a real number between zero and one. Also, larger the difference larger is the preference. Six basic types of preference functions have been defined, for more information see [184].

$$\begin{gathered} P_j(a,b) = F_j[g_j(a) - g_j(b)] \\ 0 \leq P_j(a,b) \leq 1 \\ j = 1, \ldots, m \end{gathered} \tag{4.11}$$

PROMETHEE procedure uses pairwise comparisons, which allows the evaluation of each solution. The preference between two solutions, taking into consideration all the criteria, is expressed by Equation 4.12. $\pi(a,b)$ gives the degree in which solution a is preferred to solution b, and $\pi(b,a)$ how solution b is preferred to a.

$$\begin{cases} \pi(a,b) = \sum_{j=1}^{m} P_j(a,b) w_j, \\ \pi(b,a) = \sum_{j=1}^{m} P_j(b,a) w_j \end{cases} \tag{4.12}$$

Properties presented in Equation 4.13 hold for all pair of solutions (a,b). If $\pi(a,b) \sim 0$ it indicates a weak global preference of solution a over solution b. On the other hand, if $\pi(a,b) \sim 1$ it indicates a strong global preference of solution a over b.

$$\begin{cases} \pi(a,a) = 0, \\ 0 \leq \pi(a,b) \leq 1, \\ 0 \leq \pi(b,a) \leq 1, \\ 0 \leq \pi(a,b) + \pi(b,a) \leq 1 \end{cases} \tag{4.13}$$

Calculating $\pi(a,b)$ and $\pi(b,a)$ for each solutions pair is obtained a complete outranking valued graph, by drawing two arcs between solutions a and b. Figure 4.5 depicts an example.

Each individual solution face the competition of $(n-1)$ other solutions. The outrank of $(n-1)$ solutions from the perspective of solution a is given by $\phi^+(a)$ and $\phi^-(a)$. Positive outranking flow, $\phi^+(a)$, express how solution a outranks all the others. Higher $\phi^+(a)$

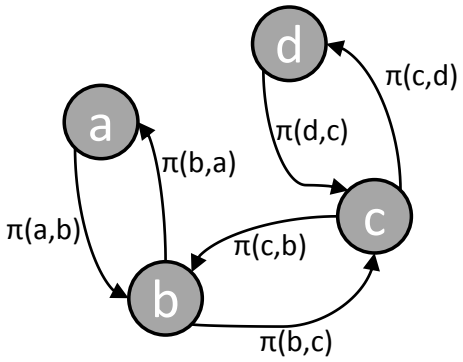


Figure 4.5: PROMETHEE: Example of an Outranking Valued Digraph.

value, better is the solution. Negative outranking flow, $\phi^-(a)$, express how solution a is outranked by all the others. Lower $\phi^-(a)$ value indicates better solution. $\phi^+(a)$ and $\phi^-(a)$ expressions are defined in Equation 4.14.

Consider S_{Set} as the solutions set.

$$\begin{aligned} \phi^+(a) &= \tfrac{1}{n-1} \sum_{x \in S_{Set}} \pi(a,x), \\ \phi^-(a) &= \tfrac{1}{n-1} \sum_{x \in S_{Set}} \pi(x,a) \\ a &\in S_{Set} \end{aligned} \tag{4.14}$$

The PROMETHEE procedure described above is used by all PROMETHEE variations. Each method individually applies its own extension for its particular focus. PROMETHEE methods I and IV use preference functions which are more suitable for qualitative criteria. Methods type III and V are normally chosen for quantitative criteria problems. Type II and VI preference functions are used with less frequency [179].

This method is consistent, easy to use and does not need great interaction with decision makers. Although, PROMETHEE presents as downside the incapability to react when a new alternative is introduced, and method II is unable to present the preferred solution in a bi-criteria problem, after method I had been applied.

4.2.3 ELECTRE

Elimination Et Choix Traduisant la Realite (ELECTRE — Elimination and Choice Expressing the Reality) method was created by Bernard Roy [191]. Similar to the previous decision method, ELECTRE has also six variations: type I, II, III, Iv, IS and TRI.

ELECTRE is a preference-based model, performing a pair-wise comparison between solutions. The model is based on three types of relations between solutions: is preferred to; is indifferent to; and incomparable to. From these, ELECTRE methods use binary outranking relations, S, to characterize four possible situations which can occur between two solutions (a and b), as Equations 4.15 shows:

$$\begin{gathered} aSb \text{ and } (not\ bSa), \text{ i.e.: } aPb\ (a \text{ is strictly preferred to } b), \\ bSa \text{ and } (not\ aSb), \text{ i.e.: } bPa\ (b \text{ is strictly preferred to } a), \\ aSb \text{ and } bSa, \text{ i.e.: } aIb\ (a \text{ is indifferent to } b), \\ (not\ aSb) \text{ and } (not\ bSa), \text{ i.e.: } aRb\ (a \text{ is incomparable to } b) \end{gathered} \tag{4.15}$$

To define the outranking relations, S, two major conditions must be satisfied to validate a relation aSb. The conditions are: $Concordance$ and $Non–Discordance$. $Concordance$ condition states that for an assertion aSb to be valid, a sufficient number of criteria must support this claim. Meaning that for a solution a to be strictly preferred to b, solution a must be better in a majority of criteria. $Non-Discordance$ condition states that if $Concordance$ condition is true, none of the criteria in minority set should oppose too strongly to outranking relation aSb [184].

ELECTRE methods also bring two more notions, the criteria importance and veto thresholds. Criteria importance is the "weight" given to a criteria j, i.e.: w_j. It is used to reflect criteria power in the majority set in favour for a certain outranking relation. These weights do not depend from criteria types, ranges or values. Veto threshold is a value, affiliated to each criterion, which reflects the possible difference between two solutions for that specific criterion. These values can be constant or vary.

All ELECTRE methods are based on the same background, but have different processes. In ELECTRE I and IS, a single solution or a group of solutions are selected and assigned as a possible solution. ELECTRE II was developed to deal with problems that need to rank solutions from the best to the worst solution. ELECTRE III introduced pseudo-criteria and fuzzy binary outranking relations. ELECTRE Iv method was possible to rank solutions without the use of relative criteria coefficients (the only ELECTRE method that did not use it). Finally, ELECTRE TRI assigns categories to solutions [179, 187].

Generally, ELECTRE methods do not frequently lead to the case in which one solution stands out from the others. For this reason the method is considered to be more suitable for problems with several solutions and not so many criteria.

4.3 Topic Discussion

Looking to the thesis scope and identified problem, is possible to see that today we are living in a world with a large number of heterogeneous devices, with capabilities to implement an uncountable number of tasks. Consequently, it is very important to develop and provide methods capable to assist engineers on the selection of a proper, more suitable hardware platform (device) to implement such tasks or systems.

Taking into account that Multi-Criteria Decision-Making (MCDM) is one of the most widely used decision methodologies in sciences, business, governmental and engineering worlds. This section has presented a literature review on well-known MCDM methods, capable of improving the quality of decisions made by decision makers, for example, in

choosing a platform from a set of available hardware alternatives. Reviewed methods are focus on problems with finite and known number of solutions/alternatives.

The MCDM method, Analytic Hierarchy Process (AHP), is commonly applied to problems with multiple, and even conflicting criteria. AHP classify solutions, pointing out the proper one with judgement based on criteria. This method presents as main advantage, compared to the other two methods, the capability to decompose a problem into small parts, becoming clearer the importance of each criteria. Although, it is not suitable for cases were the large number of criteria is maintained. The pair-wise criteria comparison becomes difficult, and the use of the nine-point scale is also a limitation. For example, a criteria k which is more important than a criteria r four times, and criteria r is also four times more important than a criteria s. AHP method presents a serious evaluation problem to deal with the scenario where a criteria is sixteen times (16x) more important than another [190]. Table 4.2 points out other strengths and weaknesses of this method, as for PROMETHEE and ELECTRE methods.

Table 4.2: AHP, PROMETHEE and ELECTRE: Main Advantages and Disadvantages.

	AHP:	**PROMETHEE:**	**ELECTRE:**
Advantages:	• Flexible and checks inconsistencies; • Gathers subjective and objective evaluation measures; • Able to assist in group decision-making; • Decomposition of a problem;	• Normalisation is not needed; • Able to assist in group decision-making; • Easy to use;	• Takes into consideration uncertainty; • Able to deal with qualitative and quantitative criteria;
Disadvantages:	• Unfit for large number of criteria; • Important information may be lost; • Criteria weights have direct impact on the result; • Nine-point scale limitation;	• Do not structure the problem; • Difficult to use with many criteria and alternatives; • Not a single method;	• Difficult to understand: Find concordance and discordance matrix; • Definition of thresholds: no "correct" value; • Time consuming;

PROMETHEE, the second MCDM method addressed in this section, is an outranking method that sort solutions based on criteria and decision maker preferences. Comparing with AHP, PROMETHEE needs much less inputs, only solutions evaluation through criteria must be performed [190]. It is considered a strong methodology basically due to its mathematical properties and ease of use [184]. To use PROMETHEE it is necessary to choose a preference function, and it has to be selected carefully by the decision maker, which can turn out difficult for an inexperienced user. Another downside is the lack of problem structuring, to give a more clear view of the problem.

The last aimed MCDM method is ELECTRE. This method generates binary outranking relations between alternatives, based on a concordance and discordance index. It is capable to handle qualitative and quantitative data with high uncertainty. It has a good performance when decision makers are not able to give rational information about some aspects [180]. ELECTRE do not use pairwise comparison, and is more suitable in cases with a large number of solutions [181]. It is also less sensitive to addition of other solutions, in comparison with the previous method.

Chapter 5

Framework to Formally Describe an IoT System

This chapter intends to present the author's first conceptual contribution, which is a framework for IoT Systems formal description. Sub-research question Q1.1 presented in Section 1.4.1 refer the lack of methods to formally describe an IoT System. But before the framework presentation, a background study, is presented in next section, on how IoT Systems are described and in which levels, areas and the detail it is done.

5.1 Models, Methods for IoT Systems Specification

Literature has been more focus on functional/behaviour aspects, and in activities/actions/interactions within an IoT Deployment [16–18]. Internet-of-Things (IoT) is addressed from an observation/functional point of view and not to fully describe an IoT System. The proof of this is the creation of standards such as the Systems Modeling Language (SysML) [19, 20] from the Object Management Group (OMG), the Sensor Model Language (Sensor Model Language (SensorML)) [21] from the Open Geospatial Consortium (OGC) and Semantic Sensor Network (SSN) [22] from the World Wide Web Consortium (W3C).

Standards have also been proposed to define systems and systems-of-systems, modelling requirements, behaviours, processes, etc., supporting interoperability at a syntactic as semantic level. One example is the OMG' Systems Modelling Language (SysML) [19, 20, 192], a general-purpose architecture modelling language for engineering systems. SysML is an extension to the OMG' Unified Modeling Language (UML), designed to support the specification of requirements, structure and behaviour, as verification and engineering systems validation. Interoperability between engineering tools is supported by the use of OMG XMI 2 model interchange and the ISO 10303 STEP AP233 data interchange standard. SysML also supports the use of the OMG's Model Driven Architecture

(MDA) techniques (see Section 3.2.2).

The Sensor Model Language (SensorML) [21], an OGC approved standard, main objective is to enable interoperability, syntactic as semantic using ontologies and semantic mediation. SensorML represent components, physical (e.g. detectors, actuators) and non-physical (e.g. mathematical operations or functions), as processes. Sensors and sensor systems are defined using geometric, dynamic, and observational characteristics. SensorML describes sensors functional models, although it states that can, but generally does not, provide detailed information of a sensor hardware design.

Other example is the W3C Semantic Sensor Network (SSN) [22], an ontology to describe sensors and observations. SSN was developed with two main objectives, first create ontologies to describe sensors, and second provide an extension (semantic annotations) to the SensorML. The created ontologies allow classification and reasoning over sensors capabilities and measurements. Sensor and services semantic annotations allow data to be organised, managed, queried and understood by different systems.

5.1.1 Hardware Representations

In the IoT domain, the formal representation of full hardware system, platform has been some left behind. Frameworks, architectures describe IoT hardware with a level of precision as needed. Next is presented a background study on how hardware components are represented and with what level of detail it is done. Main hardware components are identified to support the proposed hardware specification of an IoT System.

The work presented in [193] identifies four central hardware components for a sensor node, the processing, power, transceiver and sensing units. The processing unit responsible for managing all procedures can also contain a sub-component to storage data. Power unit, the more important unit, i.e. no-energy, no-work. The transceiver used to communicate with the rest of the network. To sense the environment, the sensing unit, to which is recognize the existence of two sub-components, the sensor and ADC units.

The publication [43] reports to a so called "typical" representation of a sensor node architecture, containing six components: the micro-controller, battery, ADC, sensing unit, external memory and a transceiver. The work developed in [11] states that sensor nodes are equipped with processor, memory, power supply, radio, an actuator and sensors.

In [194] it was identified eight components: processing unit, memory, power supply, security, sensor, actuator, machine-machine and human-machine interface. It is stated that the presented structure is the maximum composition possible.

The literature review regarding the hardware components of an IoT System presents clearly an issue, already identified by the author, which is the lack of a formal representation to specify the hardware components and their characteristics of an IoT System. All the works analysed identify the core components, but through graphical representations. Figure 5.1 depicts, also in a graphical form, the four representations of sensor node architecture.

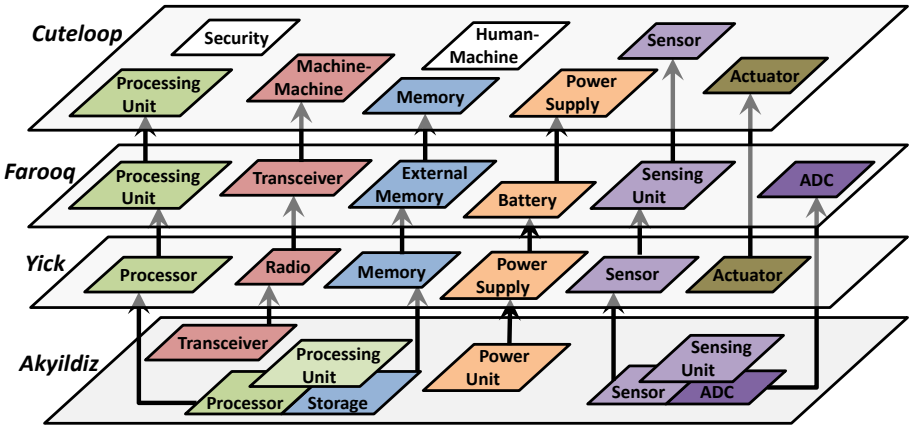


Figure 5.1: Hardware Components of a Sensor Node.

The authors Akyildiz [193], Farooq [43] and Yick [11] share almost the same idea on which the core components are. The slightest difference is focused on two specific components. First, the case of component "memory" belongs or not to a processing unit, i.e. if component memory is a core or sub-component of a sensor node. The second case is similar, and is referred to component sensing unit and its sub-components sensor and ADC.

In Cuteloop [194], a European Project of the 7th Framework Programme — "Cuteloop: Customer in the loop: using networked devices enabled intelligence for proactive customers integration of the drivers integrated enterprise", the authors presented three new core components for the architecture of a sensor node in comparison with the other publications. The new components are security, machine-machine (replacing "transceiver", "radio") and human-machine interfaces.

The author considerations will be presented in Section 5.4.1 and consequently propose a method to formally specify the hardware of an IoT System and its characteristics.

5.2 Framework for IoT System Formal Description

The Internet-of-Things (IoT) is here to stay: IoT deployments abound, more IoT-related technologies and new IoT apps are launched each day. We live in a highly connected world, connected to a large number of things making us more aware of the surrounding environment, consequently more proactive and less reactive [7]. IoT smart "things" have been playing an important role and will keep doing it in up-coming years with novel and diverse business models [3, 32, 33].

Markets are offering a wide device diversity, with a very specific characteristic — Resource-Constrained. Manufacturers are engaged in developing new embedded systems for different purposes to address the variety of application domains and services. This factor enhances even more the established heterogeneous nature of IoT Deployments.

Some manufacturers provide in their websites ways to select a product from a set

of choices. Products are displayed with some detailed (number of features), but limited to hardware characteristics. Interaction is completely handmade, with no way to apply user's preferences between criteria. Information data, when possible, can be collected only by user's action and change depending on the performed query. Two examples can be found in [14, 15].

Consequently, methods to formally describe these imperative pieces (IoT Systems) are needed, independent of hardware platforms, as well as independent from programming languages — Platform Independent. Also important is the capacity to be used by applications in an automatic way. These factors, hardware and software code, pose as core aspects to properly describe an IoT System.

Section 1.2 referred that literature involving IoT, specifically Resource-Constrained Systems (RCS), should be study carefully. Methodologies, methods analysed, and key features/aspects to describe RCS identified. Chapter 2 focused on this theme.

To formally describe IoT Systems, a key approach was identified — Model-Driven Engineering (MDE). This is due to the fact that model-driven approaches revealed to be a common ground in literature [19–23], regarding for example information modelling and interoperability. This topic was addressed in Chapter 3.

To overcome the identified issue, lack of methods to formally describe an IoT System, the author proposes the framework depicted in Figure 5.2. The proposed framework is composed by three main blocks, *Harmonisation Engine*, *Storage* and *IoT System*. The description of the considered blocks is the following:

- **Harmonisation Engine:** Responsible for parse information and mapping to IoT System specifications. This block takes use of the MDA techniques to fulfil its goal, by merging or transform the raw information. Ontologies can be used to identify and reasoning over data gathered. For example, matching, conversion of units can be accomplished with this semantic analysis. This theme will be addressed in more detail in Section 5.5;

- **Storage:** Divided in two sections, section Formalisms/Specification, serves as repository for IoT System specification models (meta-models), and rules that execute data interchange (transformations). At this level, IoT System Meta-Models form a representation of a generic IoT System. Referring to MDA layers, this will be located at abstraction level 2 — PIM (see Section 3.2.3). The second one, section Data, serves as repository for data models that are created and consequently have to be stored along the transformation process. At this level, a representation of a specific IoT System is created, i.e. structure data/information is stored describing an IoT System. Relatively to MDA layers, a representation of a specific IoT System correspond to abstraction level 1 — PSM;

- **IoT System:** Represents a full description of an IoT System, following the established specifications and relations. An IoT System, if it does not already exist in the

Storage block, can be defined from already available specifications (e.g.: combine different hardware parts). Another possibility is the generation of a new IoT System representation as a result of parse information and mapping from block *Harmonisation Engine*. An IoT System is specified by a hardware model, a software model and a generic features specification model. It is also possible to include an energy profile model, but is not mandatory. The energy profile is based on both software and hardware information.

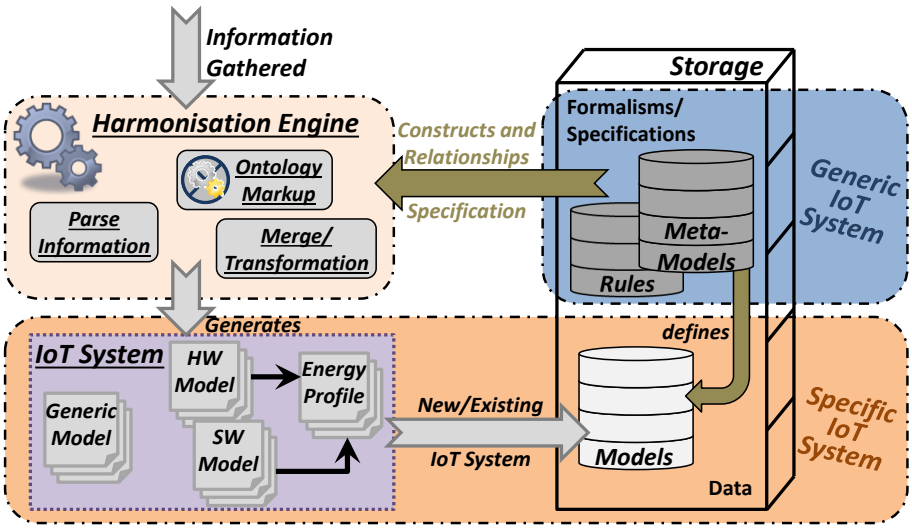


Figure 5.2: A Framework to Formally Describe an IoT System.

Next Sections will address the specification models that describe an IoT System. First is presented a specification for generic features/properties, their domains and units. Followed by the IoT System specification, that includes hardware meta-model, examples of software specifications and energy profiles specification.

5.3 Specification of IoT System Generic Features

Prior to present the IoT System specification, is necessary to describe how properties are defined by all modules of the presented framework.

IoT Systems Analysis Generic (IoTSAG) Meta-Model, presented in Figure 5.3 is a specification model used to instantiate properties/features/criteria and related aspects (property domains and units), in a way that consistency is maintain between framework models, as well as assist in a easy interoperability between tools.

The main model object of IoTSAG Meta-Model is the *Definitions* class. This class is responsible, mainly, for representation and aggregation of properties domains and define/specify units. IoTSAG specification also contains other important classes, two types of *Properties*, the *SystemDescription* and *Annotation*.

The class *Property* is an abstract class (i.e. cannot be instantiated), used to embrace two types of properties, *AggregationProperty* and *SingleProperty*. *Property* class exist to be

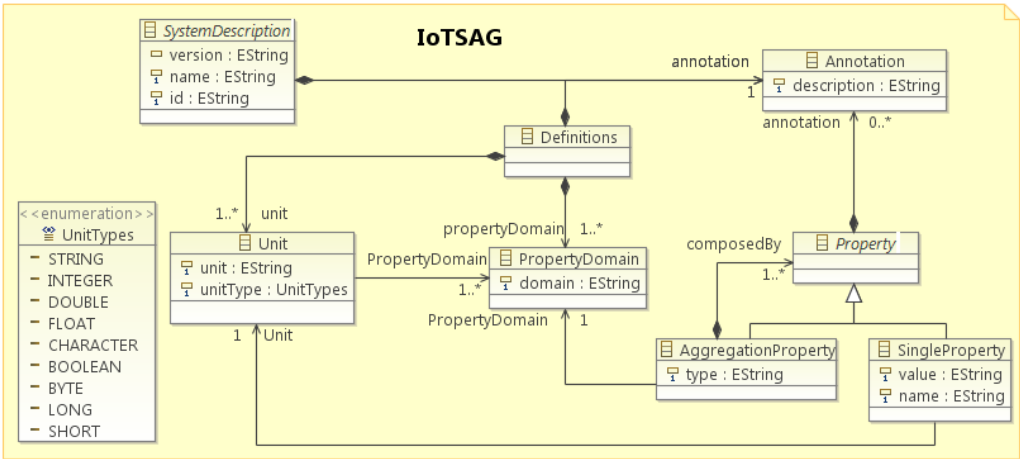


Figure 5.3: IoT Systems Analysis Generic (IoTSAG) Specification Model.

referenced by other specification models. *SingleProperty* class has the objective of define a single instance value. Composed by a name and a value, this property has also to refer to a unit type. On the other hand, *AggregationProperty* class is used to define composed properties, i.e. a property that contains more than one single value to be instantiated (e.g. communication has range, transmission rate, etc.). This class has to reference one type of domain (e.g. communication, price, weight). *AggregationProperty* and *SingleProperty* inherit from *Property* class, and it is this two classes that actually are instantiated by other models.

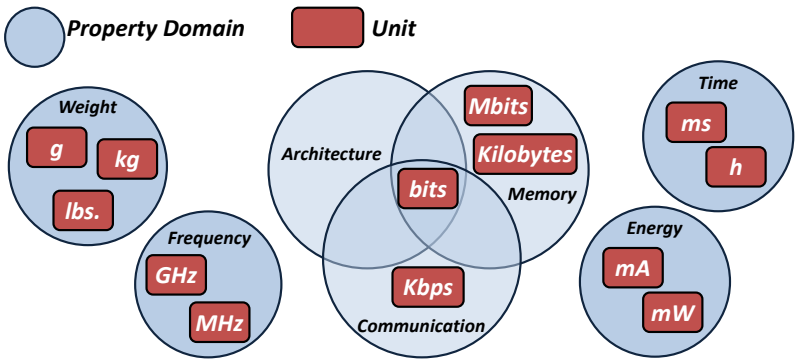


Figure 5.4: Property Domains and Units Relation: Graphical Example.

Classes *PropertyDomain* and *Unit* are responsible for maintaining a common ground, a relationship between different units of the same feature. Figure 5.4 depicts some graphical examples of well known units and their corresponding domains. Identified as property domains is weight, frequency, architecture, communication, memory, energy and time, each one embracing a set of units. For example, “g” (gram), “kg” (kilograms) and “lbs” (pounds) are units used to represent weight, or ways to represent time such as “ms” (milliseconds) and “h” (hour). However, there are cases where a unit (“bits”) is shared by different domains (architecture, memory and communication).

The *SystemDescription* class is an abstract class that contains generic attributes (version, name and id) to assist in a system description (e.g. IoT System). *Annotation* class is used to add additional information. Normally used for comments, but can also be used to describe scenarios, objectives, functions, etc.

The Ecore representation of the IoTSAG Meta-Model is presented in Appendix A.

5.4 IoT System Specification

Section 2.2.2 referred the ongoing evolution on micro-electronic technologies bringing to IoT domain a brand new series of devices/systems. These IoT Systems are in fact Resource-Constrained Systems (RCS), as result of their particular features (e.g. small size, low-power, low processing).

To formally describe these imperative pieces (IoT Systems) in a complete and conscious form some concepts should be taken into consideration. Hardware characteristics vary greatly from one device to another, therefore a hardware specification has to be comprehensive enough to embrace hardware platforms diversity. The variety of available software languages poses as a higher issue, since there is not a common ground for the rules they obey, therefore a careful analysis is needed. Finally, and mentioned as other key aspect in IoT evolution, is the energy consumption (see Section 2.1). Energy information is not always available, it is still an ongoing research and besides that when available, such information representation differ or in case of further research is unknown.

Consequently, the Resource-Constrained System Meta-Model (RCSM) (see Figure 5.5) is the proposed specification model to describe an IoT System. In the figure is also possible to see the relations with other models. An IoT System formalisation is composed by three core parts. These core parts, represented as class are the *HardwareModel*, *SoftwareModel* and *EnergyProfile*.

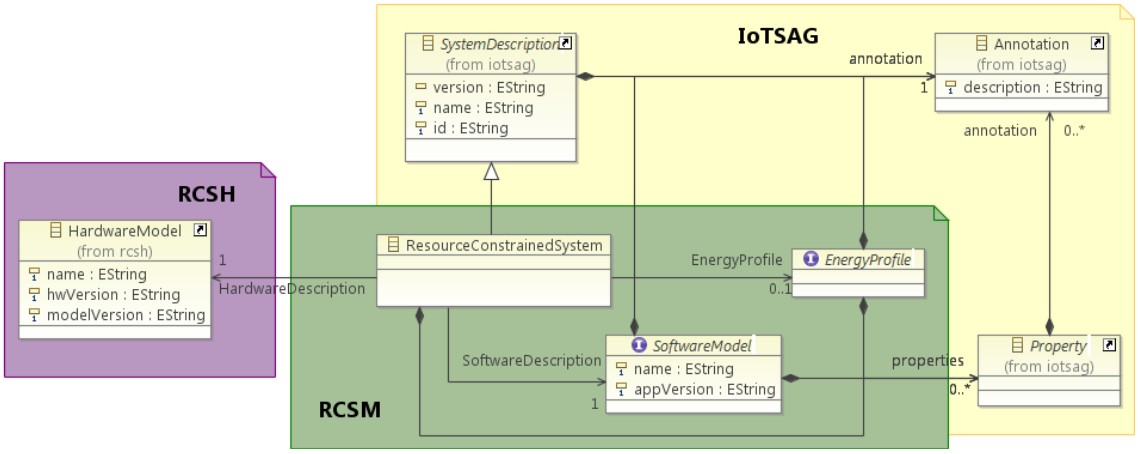


Figure 5.5: Resource-Constrained System (RCS) Specification Model.

The reasoning over an IoT System description is the following. The representation of IoT System hardware characteristics could be achieved by a single specification model

(Section 5.4.1 describes the proposed meta-model). Representation is identified through the connection to the hardware specification main model object, *HardwareModel*. Literature review in Section 2.2.2 presented some forms on how to describe a device in terms of hardware, such as [11, 43, 193, 194]. It was then possible to prepare a model to fully embrace device hardware characteristics, maintaining openness for different hardware platforms — Platform Independent.

Although, for the software languages and energy information cases, is not possible to achieve a single model. Programs follow specific rules of the software languages in which they are written from. With energy information the scenario is similarly, different forms of represent, simulate energy consumption. Consequently, a different approach is proposed. Classes *SoftwareModel* and *EnergyProfile* are in fact interface classes, allowing instantiation of different software languages and energy profile models. Furthermore, non-existing/forthcoming specification models (software and energy) can be associated to an IoT System.

In these cases, the proposed framework is not bound to a restricted, pre-established specification models. Maintaining an important feature — Platform Independent. Nevertheless, software and energy specification models have to respect two rules. Properties must be specified by IoTSAG model and the main model objects have to inherit the corresponding interface (e.g. SoftwareModel).

The three core parts are addressed in detail in the following sections.

The main model object (class *ResourceConstrainedSystem*) inherits the attributes from IoTSAG *SystemDescription* class, and more important it is able to instantiate IoTSAG properties. Features instantiation, such as cost, memory used, final physical size, implementation time/effort/difficulty, etc., are directly connected to a RCS in its final form and not to one of the parts (e.g. Hardware) that builds it.

As last consideration regarding the use of RCSM is that a device (physical part) can be associated with different software languages, forming different RCS. Or simply joining different hardware parts (shields/expand boards that can be coupled together). That is, from one hardware or one software model multiple RCS (multiple solutions) can be obtained.

The Ecore representation of the RCSM is presented in Appendix B.

5.4.1 IoT System: Hardware Specification

Unlike the programming languages, that present well defined meta-models or grammars, a hardware specification model capable of representing an IoT System is still missing. Next is presented a way to formally represent a hardware platform/device and all its hardware components, with a careful consideration for interoperability and integration with tools and systems.

The author's proposed specification model is based on the previous background study. From the presented works, it was possible to conclude that some of them are strongly

focused on functional, behaviour aspects and in activities, interactions within an IoT Deployment [16–18], rather than on IoT Systems itself. Also, the available hardware descriptions are high level, i.e graphical representations of the hardware components [11, 43, 193, 194].

Figure 5.6 identifies the core components of a hardware system in the author's perspective. It is considered that these components are adequate and no hardware component is left without representation.

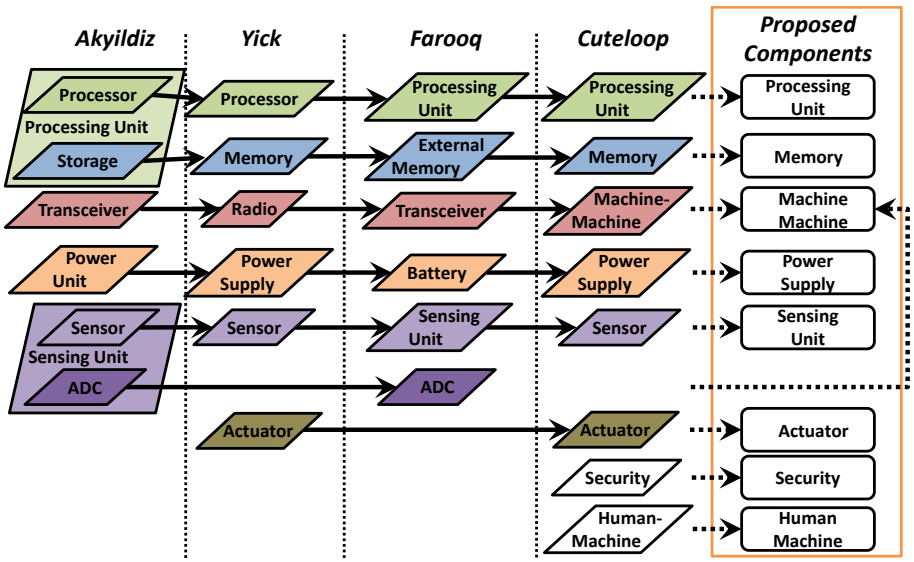


Figure 5.6: Components Considered for the Hardware Specification Model.

Most of considered components are self-explanatory, although a clarification is important regarding *SensingUnit* and *MachineMachine* components as to the cases where a component contain other components (reported by Akyildiz [193]).

It is considered by the author that a *SensingUnit* is the sensor itself (e.g. thermostat) any additional unit (normally an ADC unit) belongs to the sensing component but it is not a sensing unit. A *MachineMachine* component is any hardware part responsible for any kind of interaction between two electronic components. The ADC is an example, converting analogue data from a sensing unit to a processing unit digital port. As depicted in Figure 5.6, a transceiver is also a good example, processing the data from the processing unit to wireless waves that will be pick-up by other transceivers.

It is clear at this point that a component can be form, contain other components. For example a processing unit can have memory, ADC, M2M components. It is essential that the proposed specification has this into consideration, i.e. a component can contain other components.

To tackle the lack of formal descriptions that allow a complete specification of an hardware system, it is proposed the Resource-Constrained System Hardware (RCSH) Meta-Model depicted in Figure 5.7, one of the core parts to describe an IoT System.

RCSH specification model provides a way to formally describe a RCS in terms of its

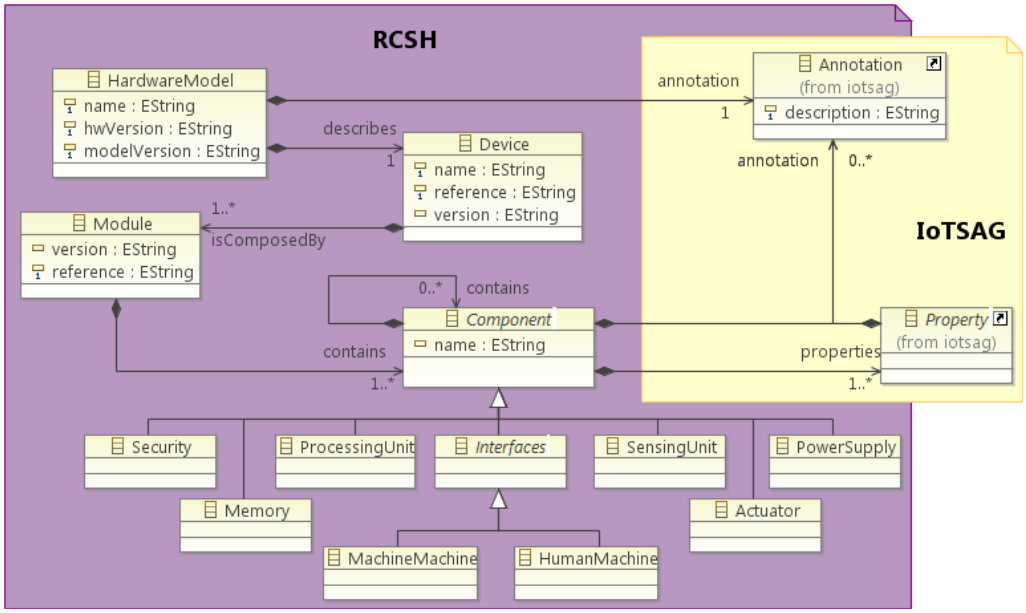


Figure 5.7: Resource-Constrained System Hardware (RCSH) Specification Model.

hardware components. A more detailed explanation of the Meta-Model composition is presented, evidencing the use of each defined structure:

- **HardwareModel:** the main model class, includes a model version (for version control) currently at "4.0−2019", a name settle as "Resource-Constrained System Hardware Model", a hardware version (to control the content of the model versus any possible change in hardware by the manufacturer), and a description (from IoTSAG meta-model). A *HardwareModel* is composed by one device (class *Device*), i.e. the physical part of a RCS;

- **Device:** represents the whole hardware of an IoT System. It is identified by a name and a reference (used by manufacturers), can also include a version. A *Device* contains one or several modules (class *Module*);

- **Module:** this class represents the important individual hardware parts of an IoT System. The *Module* class identifies by reference and version (not mandatory) each part. The use of multiple modules is in case of an IoT System built from different parts (e.g. a processor unit and a communication shield). A *Module* can contain one or more components (class *Component*);

- **Component:** is an abstract class to represent any kind of hardware part available in an IoT System. It is identified by name, and can contain other components has it was highlighted as an important feature (instantiation of other *Component* classes using "contains" reference). It can also include properties and additional descriptions from IoTSAG specification model. An instantiation of a *Component*

can be of different types: *ProcessingUnit*, *Memory*, *Security*, *SensingUnit*, *Actuator*, *PowerSupply*, and also of two types of *Interfaces*, *MachineMachine* or *HumanMachine*;

- **Interfaces:** is an abstract class to represent two types of connections. First, a Machine-to-Machine M2M interaction (*MachineMachine*), and second, a human—machine interaction (*HumanMachine*);
- **ProcessingUnit:** is a specific *Component* used to instantiate processing units. In IoT Systems these processing units are normally micro-controllers containing several other components (e.g. memory, communication, converters);
- **Memory:** is a specific *Component* used to define different types of memory (e.g. Random-Access Memory (RAM), Static Random-Access Memory (SRAM), Flash);
- **Security:** is a specific *Component* to represent hardware parts that implement specifically security control. Security appliance has its normal implementation using software code. Although, security implementation using hardware is being more used due to its faster handling compared to software implementation. Its common hardware implementation uses programmable logic components, reconfigurable hardware;
- **SensingUnit:** is a specific *Component* to define sensor units, such as temperature, humidity, light sensors;
- **Actuator:** is a specific *Component* to define for example magnetic locking mechanisms, relays controlling water, light mechanisms;
- **PowerSupply:** is a specific *Component*, which due to the specific nature of an IoT System, is used to describe a battery and the corresponding power controllers;
- **MachineMachine:** is a specific *Interface*, which inherits *Component* features and is used to represent M2M interaction, for example communication type (e.g. Universal Asynchronous Receiver/Transmitter (UART), Serial Peripheral Interface (SPI)), or ADC converters;
- **HumanMachine:** is a specific *Interface*, which inherits *Component* features and is used to represent the interaction between human and machine (e.g. a display, led, button).

All classes inheriting from *Component* class, can instantiate properties through the reference link *properties* and can include a description using reference link *annotation* (an instantiation of class *Annotation* from IoTSAG specification model).

The Ecore representation of RCSH is presented in Appendix C.

Figure 5.8 illustrates a high level view of a Resource-Constrained System RCS being represented by the proposed RCSH specification model. This example shows an electronic

development platform (the Arduino Uno Revision 3) on the left and on the right side the corresponding formal representation of some parts that compose the device.

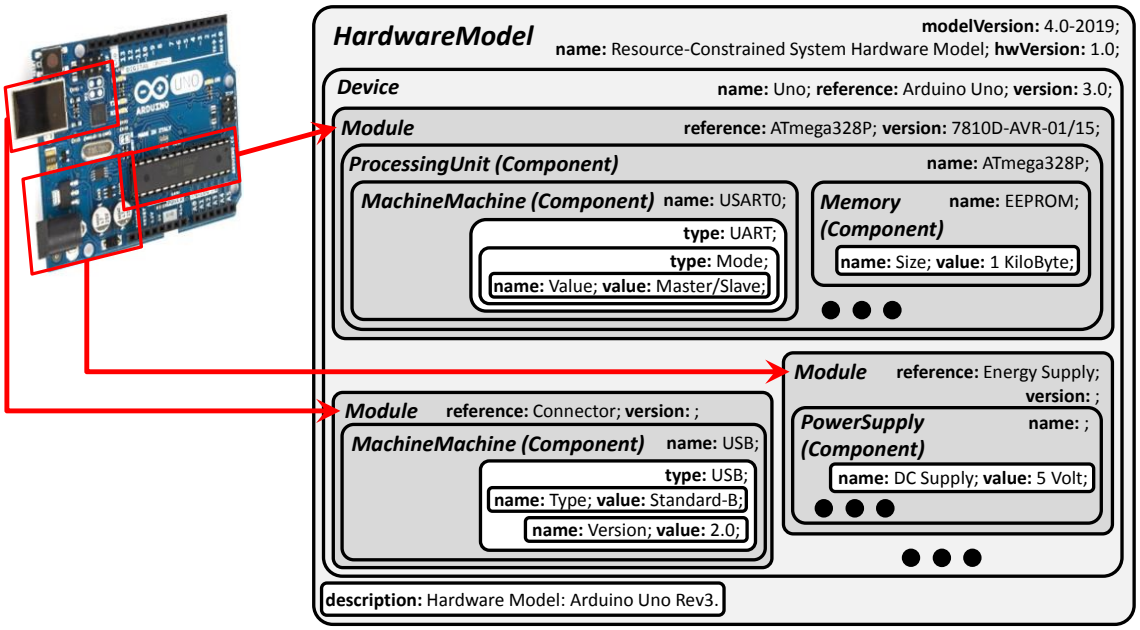


Figure 5.8: Hardware Platform: A RCSH Formal Representation Example.

The identification, representation of all parts that form a hardware device is extensive set of data. Therefore, the example identify 3 hardware parts, the micro-controller, the Universal Serial Bus (USB) connector (used to program the device) and the power supply set, and shows partial information of each one.

White boxes reports to model classes from IoTSAG specification model. For more detail regarding units and property domains specification see Section 5.3 and practical example depicted in Figure 5.4.

5.4.2 IoT System: Software Formalisation

One of the three core parts that describe an IoT System is its program, the software code. However, the number of software languages available is high and in most cases they do not share a common ground which poses as a problem. To tackle this issue the proposed framework provides an interface, so users can integrate their own software languages and consequently build a properly formalisation of their IoT Systems. Section 2.3.4 point out some programming languages known to be used in IoT (C language and nesC), but others could be mentioned (e.g. VHDL).

Describe an IoT System using the proposed framework requires an application, firmware model to be added to the RCSM. Such description, application model, must follow a specification model — Meta-Model. In case stakeholders intend to form an IoT System using a new software language is necessary to properly integrate the new application specification model into the IoT System Formal Description Framework.

To properly integrate a software language into the proposed framework, two rules must be obeyed by the specification model. Additional information, properties must be specified using IoTSAG specification and the main model object inherit from the interface class *SoftwareModel*. Figure 5.9 illustrates an example on how a software language specification model must be defined.

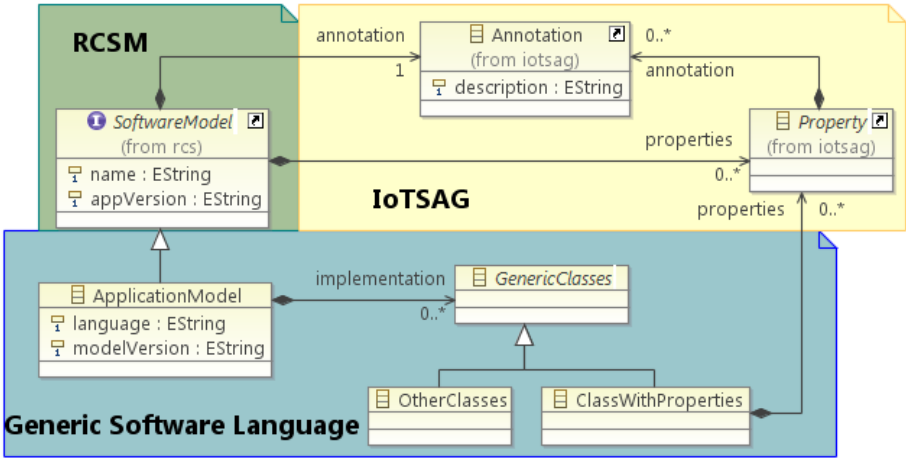


Figure 5.9: Software Language Specification: A Generic Example.

The main model object of *Generic Software Language* specification is the *ApplicationModel* class. As depicted, this class inherits, is a children class, from the interface class *SoftwareModel* of the RCSM. In this way, is enabled its use as a software language specification model, consequently as one of the parts that describes a RCS. Inheriting from *SoftwareModel* class, the main model object can instantiate properties through the reference link *properties* and include a description using reference link *annotation* (an instantiation of class *Annotation* from IoTSAG specification model). Possible properties examples for software languages could be references to messages payload or maximum number of exchange messages per day in a communication scenario, or storage limits (e.g. vectors, arrays) in a data storage scenario. These aspects, restrictions are imposed by stakeholders or systems restrictions, but applied by programmers in program codes. Even though, properties instantiation are automatically available by inheriting from class *SoftwareModel*, it is possible to other classes to reference IoTSAG properties. To exemplify this case it was included in the figure the class *ClassWithProperties*.

The example also suggests the use of two attributes (use not mandatory), *language* and *modelVersion*, for the software language main model object. The *language* attribute could specify the software language (e.g. C, nesC). With language identification additional functionalities could be released, activated such as a high level, graphical tool to show the application. The use of attribute *modelVersion* is intended for model version control, consequently versions compatibility.

5.4.3 IoT System: Energy Profile Formalisation

Unlike the two previous cases, a specification of an energy profile is not mandatory in the proposed framework, since it is not a binding factor to correctly describe an IoT System. An energy profile is an additional set of information, but not a required feature for an IoT System to work.

Nevertheless, an accurate analysis of power consumption is a fundamental support for research and development activities, since it is instrumental to predict expected IoT Systems lifetime and to allow developers to optimize energy consumption in distributed IoT applications. Two primary approaches have been followed up to now. Direct measurement is performed by engineering the devices to measure energy consumption while they are executing their distributed applications; this approach is accurate but it is very expensive since it involves engineering every single device involved, and it is not practical in large distributed applications. The second approach is related to simulation models, which are more practical and scale better, but which depend on direct measurement over smaller scenarios to collect the parameters used to enhance the realism of the model.

It is hard for a programmer to have complete knowledge of the energy consumption of his applications, since many low level details are hidden by the OS, and the details of the operations executed by the OS are device-dependent [195].

As proposed by framework in Figure 5.2, an energy profile must be built based on the hardware and application code models. These last two, together form a System Under Study (SUS), i.e. an IoT System from which is intended an energy consumption analysis.

A hardware platform can be used for several kinds of purposes, such as sensing the environment, communicate, act, process data, etc. Each one of these functions has different energy consumption impact. Furthermore, sensing a room light or temperature have different consumptions, in sending a message the consumption will be related to the amount of data that needs to be transmitted. Software code, program gives this additional, specific information necessary to properly generate the energy profile for a certain task.

Consequently, an energy consumption profile needs to embrace different aspects where time (e.g. wake, sleep time) is an important one but not unique, e.g. the particularity of the hardware components used, user code are of major importance as well. Real-life tests and available simulation methods and tools should also be analysed. Figure 5.10 proposes a high-level IDEF-0 [196] "black box" view on the energy analysis activity specifying inputs, outputs, control, and mechanisms:

- **Inputs:** the IoT System, expressed through the hardware and application model, is the system input. Along with the control definitions, they feed the energy analysis system with the information required to calculate the outputs via the application of the indicated mechanisms.

- **Controls:** variables that when adjusted can fine tune the outputs. For the specific

case of energy analysis, changing the desired degree of information (e.g. overall system or a specific function) and time constraints, influence directly the values obtained.

- **Mechanisms:** normally it is a person, a facility or a machine that executes the process. Since the process involves simulation, it requires models and simulation methods that can assist in producing the required outputs, using the information provided by the inputs and definitions by the controls. To analyse, simulate the SUS, one needs to have a well-defined SUS model. In this case, the IoT System is represented by two models, the hardware and application models, from which the system extensive composition is mitigated. Reinforced with information from available simulation methods, tools and models aggregated with real tests data, energy consumption can be extrapolated for the SUS, for different periods of time and with details for the amount of energy consumed for a specific hardware component or code function.
- **Outputs:** the system output is the energy consumption profile of the IoT System under study.

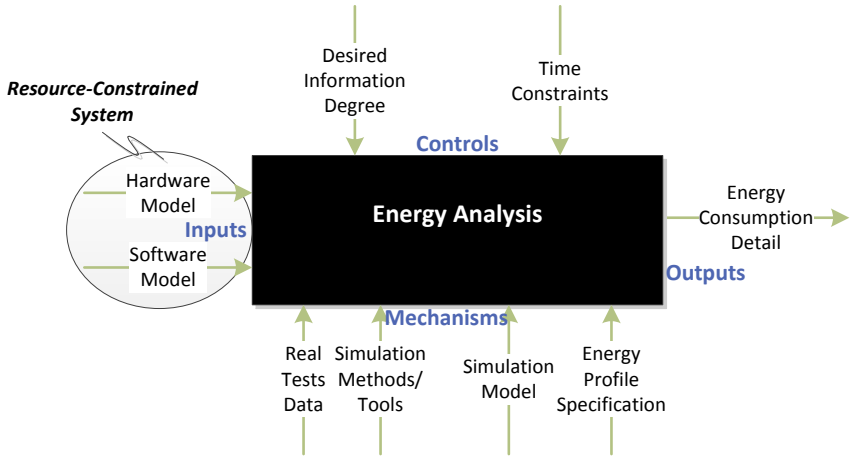


Figure 5.10: IoT System' Energy Consumption Analysis Activity Detail.

As in the software language case, the energy profile specification model has to respect the same rules so it can be properly integrated into the proposed framework. Properties must be specified using IoTSAG specification and the main model object inherit interface class *EnergyProfile*. Figure 5.11 illustrates an example on how an energy profile specification model should be defined.

The main model object of *Generic Energy Profile* specification model is the *EnergyModel* class. As depicted, this class inherits, is a children class, of the interface class *EnergyProfile* from the RCSM. This enables the association of an energy consumption model to a RCS. *EnergyModel* class automatically inherits the capability to instantiate properties (from IoTSAG) and to include a description (*Annotation* class instantiation, also from IoTSAG).

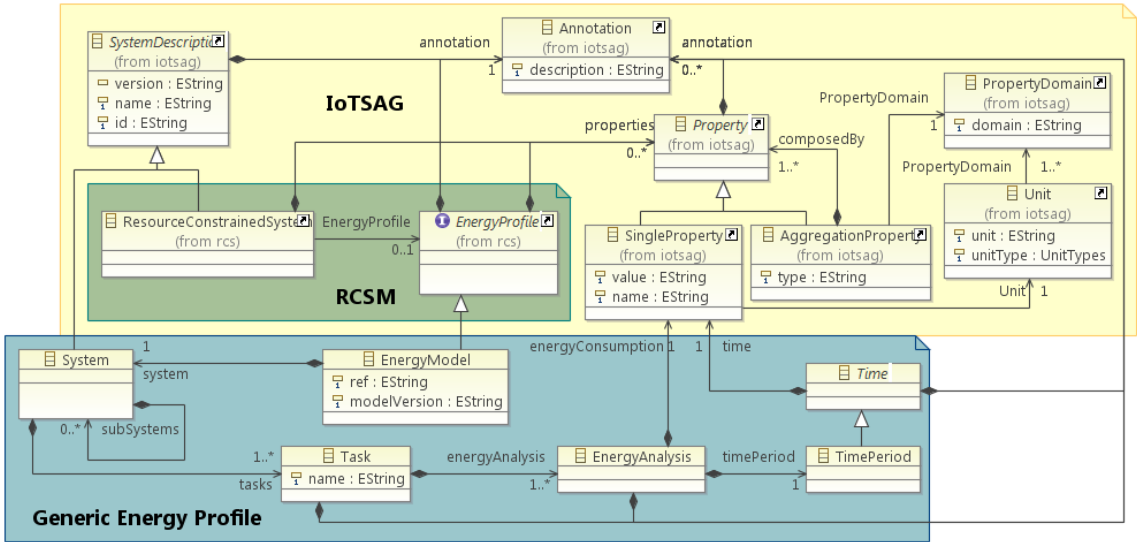


Figure 5.11: Energy Profile Specification: An Example.

The example given in Figure 5.11 shows a new specification model but an existing simulation model, once properly adapted to the two rules (properties instantiation and core class inheritance) imposed by the framework, can also be used. However, the example can be used as background for further, future developments in this area.

Concerning the specification model example to represent energy information, the reasoning used considers the representation of systems — expressing an objective of the IoT System, tasks, and energy consumption. Nevertheless, other aspects may need to be introduced in the final version of a energy profile specification model. A more detailed explanation of the composition of the meta-model example for the energy profile is presented, evidencing the use of each structure defined:

- **EnergyModel:** the main model class includes a model version (for version control) and reference identifier for the RCSM owner of the interface class *EnergyProfile*, parent of this class. An *EnergyModel* is composed by one *System*;
- **System:** here the word "system" is used to identify the IoT System objective/function/job. Inherits all features from IoTSAG *SystemDescription* class: the system version, name, identifier and a description. It can be composed by a multitude of *System* (i.e. sub-systems of the *System*) and at least one *Task*;
- **Task:** allows a system operation to be split into multiple parts, allowing the identification where the energy is consumed with more detail. It is identified by a name, can include a description (from IoTSAG Meta-Model), and contains at least one energy analysis (*EnergyAnalysis* class).
- **EnergyAnalysis:** describes the energy consumption of a *Task*, aggregating the energy value (reference "energyConsumption" to a *SingleProperty*) with the time in which it was consumed (*TimePeriod*);

- **Time:** is an abstract class to represent time aspects related to the conditions in which the energy tests or simulation take place. At this point it can be instantiated as time period (*TimePeriod*). This class is composed by a property (*SingleProperty*), from IoTSAG specification, to instantiate the time value and it can include a description (from IoTSAG *Annotation* class);
- **TimePeriod:** is a class to represent the time window in which the energy consumption tests, simulation take place. Inherits all features from class *Time*.

Classes *System* and *Task* manages the desired degree of information, being able to specify operations, code lines, expressions, etc.

As one of the mechanisms for the energy analysis activity, the presented *Generic Energy Profile* specification model could describe the energy consumption information as depicted in Figure 5.12. The example describes a device, a RCS, executing two operations: sensing the environment (sensor type is not important for the scenario) and the transmission of the real-time collected data.

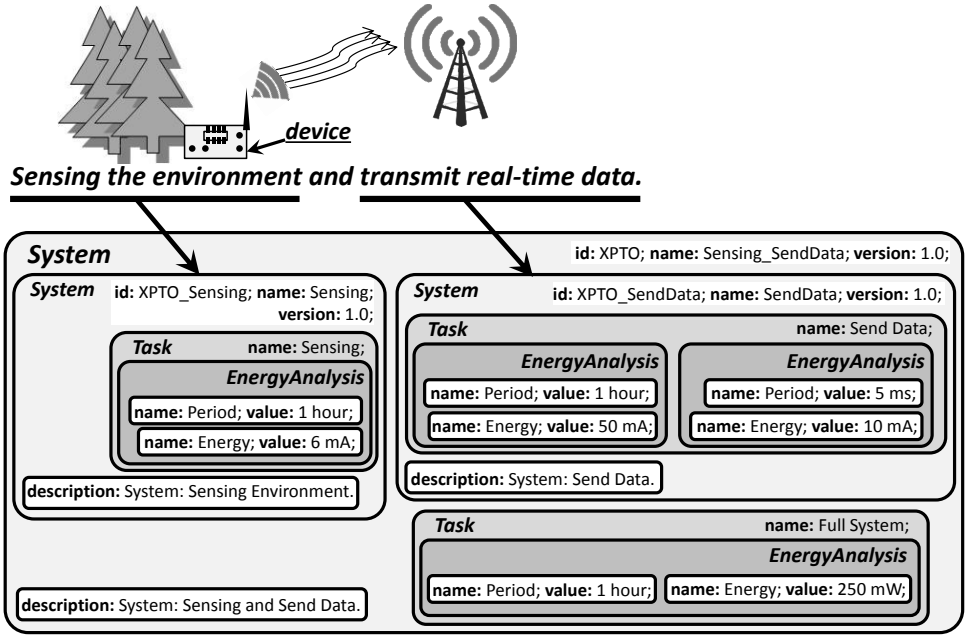


Figure 5.12: Energy Profile: A Practical Example.

The RCS is characterized as a system with "XPTO" as identifier and name "Sensing_SendData". This approach allows an energy consumption description for the full system (through the *Task — Full System*). The associated time period and energy consumption states that the full system was evaluated during one hour and had an energy consumption of 250 milliwatts. Nevertheless, and as it was described, the presented meta-model enables the evaluation of each individual operation. Operation, i.e. system "Sensing the environment", identified as "XPTO_Sensing", reports an analysis of one hour period with a consumption of 6 milliamps. On the other hand, operation "transmit real-time data", identified as "XPTO_SendData", is divided in two energy analyses for the

same task (name "Send Data"). The first analysis reports for the same time period, one hour, with a consumption of 50 milliamps. The second analysis presents a narrow period, 5 milliseconds, and consequently less energy consumption (10 milliamps).

White boxes reports to model classes from IoTSAG specification model. Different units are used for the same property domain. For more detail regarding units and property domains specification see Section 5.3 and practical example depicted in Figure 5.4.

5.5 Model-Driven Harmonisation Framework

Previous sections presented how an IoT System can be described using formalisms (specification models — Meta-Models) in terms of hardware, software languages and also associate an energy profile, guaranteeing that the IoT System representation is independent of the hardware platform and software language. The proposed specification models need a model-driven approach to materialise, support the harmonisation of the specification models, models and data in a platform-independent context. To achieve these objectives, next is presented a MDA-based framework.

Figure 3.2 in Section 3.2.1 presented a generic view of the formalisation of a System Under Study (SUS). Using the same approach on an IoT System, in this case the SUS, is obtained the representation depicted in Figure 5.13.

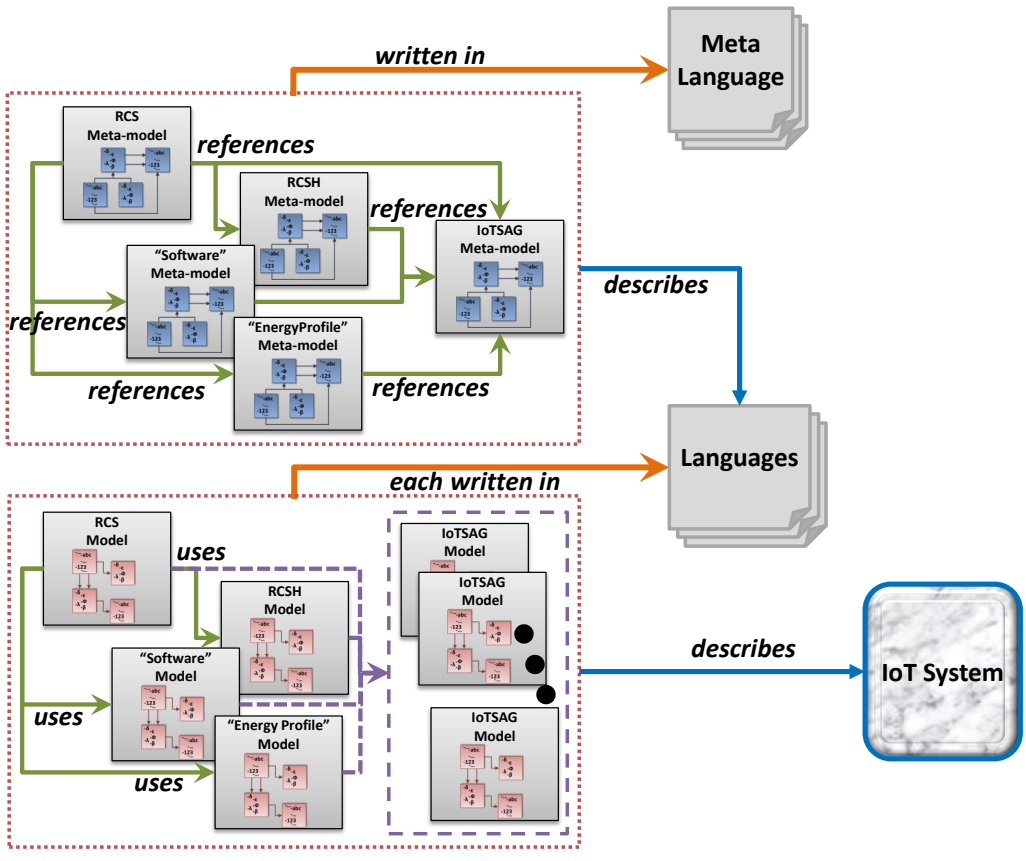


Figure 5.13: IoT System: Model and Meta-Models Relationship.

An IoT System is described by a set of models, one or more IoTSAG models, an RCSH model, a software model, and an energy profile model (not mandatory), all pinpointed by a RCS model. The use of different IoTSAG models depends of each one of the other models that describe an IoT System. Meaning that a single IoTSAG model is pinpointed by all the other models, or different IoTSAG models are referenced.

The models are written in a coherent way, syntactic as well as semantically, well-defined by the corresponding modelling language. These languages are described by Meta-Models, i.e. the proposed specification models presented in the previous sections. The Meta-Models specify the constructs and relationships within the modelling language. The proposed Meta-Models are written in the Ecore modelling language.

To address the realization of the IoT System platform-independent view a MDA-based Harmonization Framework is presented in Figure 5.14. It makes use of MDA's techniques, mainly horizontal transformations, to support interoperability between modelling languages, models and data, and streamline the integration process with other systems and tools.

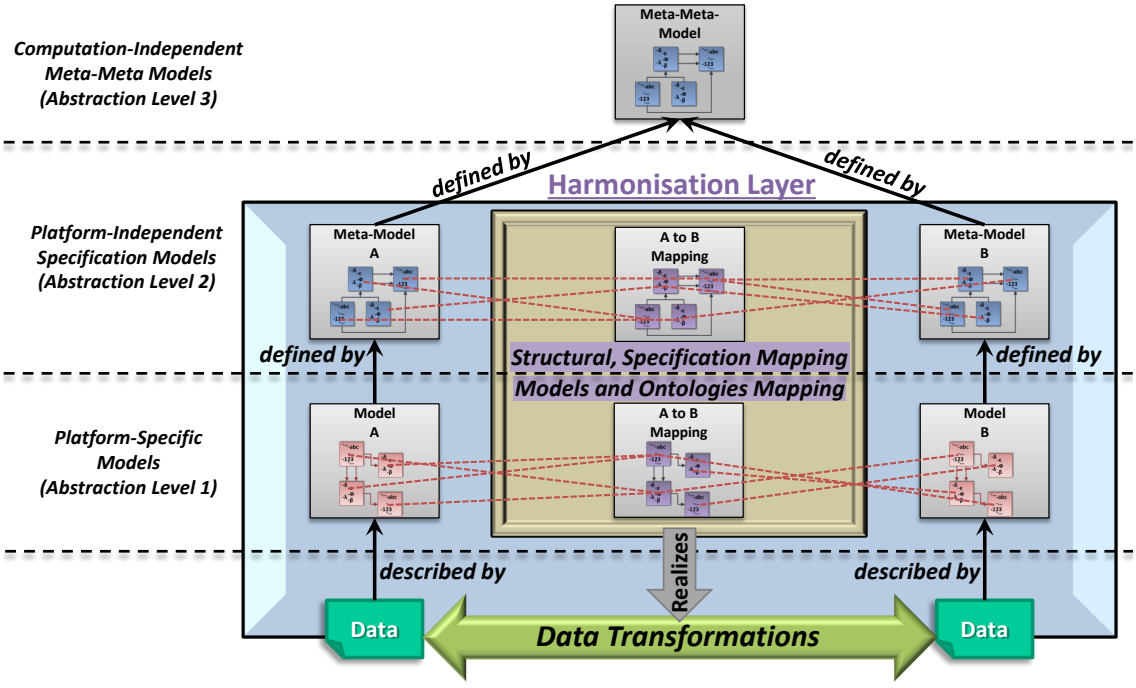


Figure 5.14: Model-Driven Harmonisation Framework.

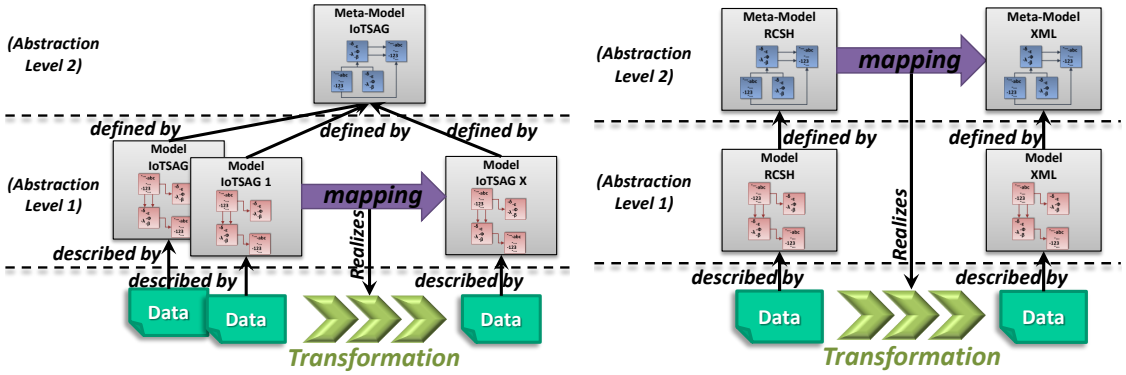
This MDA-based Harmonization Framework implementation architecture is used by the *Harmonisation Engine* of the proposed *Framework to Formally Describe an IoT System* presented in Section 5.2. It is based on the work by Agostinho et al. [197] and on the author work developed in [198], which addresses interoperability problems associated to model languages transformations, and furthermore allowing communities to build interoperable systems and services.

The left and right-hand sides of Figure 5.14 represent two specifications for information representation formats, each one with its own internal models, where information is presented following a model-language—meta-model hierarchy introduced previously in Section 3.2.1. Both representation formats can be the same, originating for example a data merge transformation, or of different formats in which the data from one-end is transformed to the representation format of the other-end.

In the middle, the *Harmonisation Layer* is responsible for model and semantics harmonisation, defining mapping morphisms among the different specification formats. Mappings are expected to be pre-defined and transformation scripts relatively static, since changes in specification models are not common. Although, transformation scripts, "enablers", can be updated to execute new rules or correct existing ones. The transformation process includes the use of a storage database. Besides the common storage action (e.g. data models storing), it includes storing of model and semantics harmonisation in a communication mediator knowledge base, to grant a planned traceability to support intelligence and sustainability. For example the use of the Mediator Ontology [199], which stores morphisms according to a tuple format [198].

Storage features and semantics analysis through terminology mapping is possible but, it is not in the scope of this dissertation.

Figure 5.15 depicts two examples of possible transformations in the IoT System formal specification scope. The first example, Figure 5.15a, illustrates two IoTSAG models being merged into a single file/model. The mapping is performed at abstraction level 1, Platform-Specific layer, since all models in this transformation share the same specification model (IoTSAG), i.e. no need to define rules between specification models (Platform-Independent layer). Nevertheless, the mapping rules to perform the merge could take into consideration the *no repetition* of *Unit* or *PropertyDomain*, or proper aggregation of units under the same property domain. The requirement of matching data within the model could be achieve through ontology markup, semantic analysis.



a IoTSAG Model Files: Merge Example.

b RCSH to XML: Model Transformation Example.

Figure 5.15: Harmonisation Layer: Mapping Examples.

The second example, Figure 5.15b, presents a transformation from one format to

another. The mapping from the source to a target specification model is performed at the abstraction level 2, i.e. at Platform-Independent layer. The selected target model is XML, a common data encoding standards widely used. The source model is a RCSH model. The specification models' mapping is performed using a syntactic matching between the models objects. The process of language mapping is manual, but the language transformations are always automatic and repeatable.

5.6 Topic Discussion

Internet-of-Things (IoT) is bringing diverse and novel business models to society, and consequently manufacturers are focus on developing new embedded systems to tackle the variety of application domains and services. It is predict that will be around 30 billion wireless connected devices to IoT in the next years [4, 5]. With a very specific characteristic, Resource-Constrained, these IoT Systems carry out simple and small applications like monitor physical or environmental conditions, such as temperature, sound, pressure, etc.

To facilitate and increase effectiveness during the design of IoT Deployments, stakeholders must be able to verify IoT Systems based on all characteristics, features that these systems have to offer. Available representations of an IoT Systems for its physical components are very high level (i.e. graphical representations), neglecting completely components parameters/values [11, 43, 193, 194]. Some manufacturers provide in their websites ways to select a product from a set of choices. Products are displayed with some detailed (number of features), but the interaction is completely handmade. Information data, when possible, can be collected only by user's action and change depending on the performed query. Two examples are [14, 15].

Consequently, the lack of ways to formally describe an IoT System, capable of being used by applications in an automatic way, is an issue. To tackle this, and to meet the first sub-research question "*Which methods could be applied or develop to formally describe an IoT System (hardware, software, energy)?*", it was proposed a framework capable of formally describe an IoT System.

To describe complete an IoT System, it is considered mandatory to have a hardware and software characterisation, but the proposed framework also enables the specification of an energy profile. Although, the framework describes means to achieve this energy profile, and initial works have been developed, it is not part of the research work of thesis.

Chapter 6

Assessment Framework for IoT Systems

Questions, either simple or complex, are something that all people have to face daily. To make a correct and accurate decision depends many times on multiple criteria, which is a tough challenge for human beings [13]. Engineers have in their hands many different devices, each one built with a very specific purpose. Consequently, they are facing the difficulty to choose in a conscious way, a more suitable IoT System on how to implement and/or improve a certain task. This need grounded the formal specification of an IoT System presented in Section 5.2, where hardware, software and energy consumption characteristics are gathered to build a knowledge database. That structured information plays an important role on the developed framework for multi-criteria assessment of IoT Systems, the author's second conceptual contribution and which is presented in this chapter.

6.1 Decision-Making in IoT

Internet-of-Things (IoT) is an umbrella term for a wide range of technologies, applications and services domains. Next are addressed some examples of decision-making in IoT scope. This literature review shows that the proposed framework is a novel-framework focused on an existing problem not yet addressed; that the envisaged decision making methodologies are widely used; and that the framework openness to different MCDM methods is a disruptive approach from one-two methods' frameworks application.

The authors in [200] focus on ensure an appropriate manufacturing processes selection to be improved, updated with IoT technologies. Processes stages were analysed for IoT application based on five criteria: reliability, security, business, mobility and heterogeneity. The selection of the more suitable part of the process to implement IoT technologies was based on applying the AHP method.

In wireless communications, poses the problem of selecting which technologies best serves the IoT network scenario. The publication [201] addresses the decision-making regarding wireless communication technologies (Wi-Fi, Z-Wave, Bluetooth, ZigBee, NFC, etc.) considering four criteria as the most important: reliability, dependability, safety and security. The application of MCDM methods is due to the wide offer of wireless communication technologies with also a large variety of features.

Another scope is security of IoT services. In [138], authors combine MCDM fuzzy methods to assess different aspects and requirements during the architecture and service implementations, to assist on decision-making regarding the allocation of security assets and resources. They combine the Analytic Network Process (a generalisation of AHP; problems are modelled as networks instead of decomposed into a hierarchy) with a decision-making trial and evaluation laboratory (DEMATEL) technique (analyse criteria' cause and effect interrelationship).

In [202], the authors present the problem of selecting an IoT platform using MCDM during the design phase of IoT. Facing the common problem of a wide offer, in this case of IoT platforms, it is proposed two MCDM approaches for weight coefficients calculation. The approaches are based in a linear convolution and the second on a multiplicative convolution method.

6.2 Framework for IoT Systems Assessment

In a world with a diversity of possible solutions, IoT Systems, engineers are facing the difficulty to choose in a conscious way, the more suitable solution on how to implement and/or improve a certain task. Sub-research question Q1.2 presented in Section 1.4.1, addressing the analysis of IoT Systems, asks for "Which methods could be applied or develop to assist in IoT System assessment".

As shown, MCDM methodologies have being applied in IoT independently of the area (e.g.: security, communications, platforms, etc.). However, a concrete decision-making regarding IoT Systems is still missing. State-of-the-art approaches or are very specific (e.g.: looking for a wireless communication protocol) or very high level (e.g.: select an IoT platform). Methodologies, tools are of standalone application (i.e. works only with the designed, the initial methods), and with no process defined for constraints application rather the normal evaluation by maximisation (i.e. optimisation by which has the higher values).

To address this challenge, a framework is presented in Figure 6.1 proposing a multi-criteria assessment framework to analyse IoT Systems, capable of suggesting the more suitable IoT System to execute a certain task. The proposed framework is composed by five main blocks, *Storage*, *Objective Characterisation*, *Multi-Criteria Decision Methods*, *IoT Systems Assessment* and *Assessment Engine*. The description of the considered blocks is the following:

- **Storage:** is in all similar to the *Storage* block from the framework to formally describe an IoT System, presented in Section 5.2 and depicted in Figure 5.2. Divided in two sections, section Formalisms/Specification, serves as repository for specification models (meta-models), in this case for Multi-Criteria Analysis Meta-Model (MCAM) and MCDM Methods, and rules that execute data interchange (transformations). The section Data, serves as repository for data models that are created and consequently have to be stored along the transformation process. As referred in the framework proposed in the previous chapter, the section Data contains the structure information regarding IoT Systems. The specification model MCAM is presented next in Section 6.3, and the integration of MCDM Methods, with two proposed specification models are presented in Section 6.4;

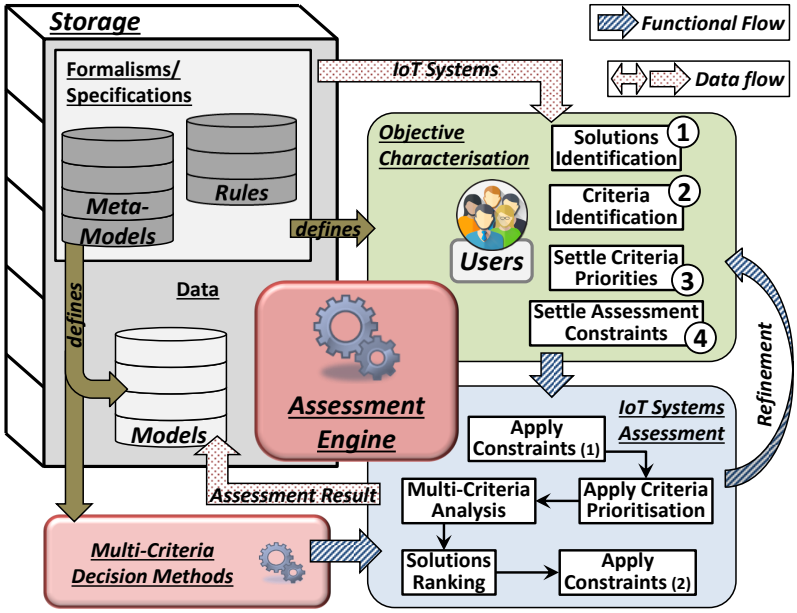


Figure 6.1: A Framework for IoT Systems Assessment.

- **Objective Characterisation:** addresses the characterisation of the problem to solve. Consists on define a purpose/objective, criteria and constraints. It is composed by four steps:

  1. *Solutions Identification:* first is necessary to identify the IoT Systems that are considered to be possible solutions capable of accomplishing the desired objective;
  2. *Criteria Identification:* decision makers choose a set of features considered important to be analysed within the objective scope. For example, and commonly selected are properties like cost, energy consumption, computation speed, etc.;
  3. *Settle Criteria Priorities:* based on decision makers judgement it is determine the preference levels between the selected criteria. Decision makers have different

perspectives of which feature is more important, and criteria importance also depends on the scenario that is being addressed;

4. *Settle Assessment Constraints:* this step allows decision makers to indicate, define some type of restrictions to criteria. It is possible to select seven types of restrictions/optimisations divided among three groups. Each constraint can be applied to one or more criteria. Two or more constraints can be applied to single feature/criterion. First, *Criteria Optimization*, is the optimization method group that allows criteria to be organised in two ways, minimize or maximize feature values. Second group is the *Criteria Availability*, split in two types. The constraint *MustHave* is used for cases in which a feature must be present, and constraint *CannotHave* for the opposite case, i.e. criterion/feature cannot be available in the final proposed solution(s) group. The third group, *Criteria Condition*, allows the definition of three different kinds of constraint levels. Constraints *LessThan*, *Equal* and *GreaterThan*, enables users to set threshold values or look for a specific feature value;

- **Multi-Criteria Decision Methods:** is the representative block on an important framework feature — openness to the use of different MCDM Methods. The proposed framework takes use of MCDM Methods to assist in the assessment process of IoT Systems, since these processes are able to evaluate diverse, contradicting criteria. Assisted by the *Assessment Engine*, in terms of specification models, transformation rules and user-defined enablers applicability, this block is able provide new and even user-defined MCDM Methods for the IoT Systems assessment process. Enablers are pieces that need to be developed and applied each time a new MCDM Method is introduced, working as interoperability facilitators between the new specification models (MCDM Methods) and the framework;

- **IoT Systems Assessment:** is the assessment procedure block. Gathered all data (following well-defined syntax) necessary to perform a conscious decision regarding the more suitable solution for a specific task, and stakeholders decided on the MCDM Method to use, is possible to execute a methodology to perform a multi-criteria analysis based on the defined constraints and criteria prioritisation, and obtain a solutions ranking. The complete assessment methodology is presented in Section 6.5. In case of assessment outcome not satisfactory for some reason (e.g. applied constraints excluded all solutions), is possible to go back for requirements refinement. For example adjust criteria conditions or constraints or even remove a criterion;

- **Assessment Engine:** is the mechanism responsible for all computation regarding the IoT Systems assessment. It carries out the data transformation from the IoT Systems specification to the MCAM, load the selected MCDM Method and respective enabler and finally execute the IoT Systems assessment procedure. The MDA-based

Harmonization Framework, presented in Section 5.5 is also applied here to accomplish the harmonisation between specification models.

Next sections will address the specification model that allows the objective characterisation by specifying criteria, constraints and possible solutions, and address the openness, dynamism of the proposed framework regarding the use of different even user-defined Multi-Criteria Decision Making (MCDM) Methods. It is presented the way to include MCDM Methods in the IoT System Assessment process. Finally, and focus on MCDM) Methods, two specification models are proposed to describe the AHP and ELECTRE methods, which were addressed in Section 4.2.

6.3 Multi-Criteria Assessment Specification

Many are the features/criteria which influence the overall performance of an application running in an IoT System. From a diversity of possible hardware solutions and software languages, a complex problem emerges for stakeholders regarding the selection of a more suitable IoT System. Combining or fulfil several requirements is not an easy task to researchers, domain practitioners and engineers [32].

Consequently, it is proposed a Multi-Criteria Analysis Meta-Model (MCAM) that formalises all parts of IoT Systems evaluation process. The MCAM is depicted in Figure 6.2. This specification model, to maintain a straightforward functionality, uses the IoTSAG specifications (presented in Section 5.3). Similarly to the RCSM, MCAM provides an interface class to allow the use of different (even new or user-defined) MCDM methods within the same problem (multi assessments by changing criteria, constraints or MCDM methods).

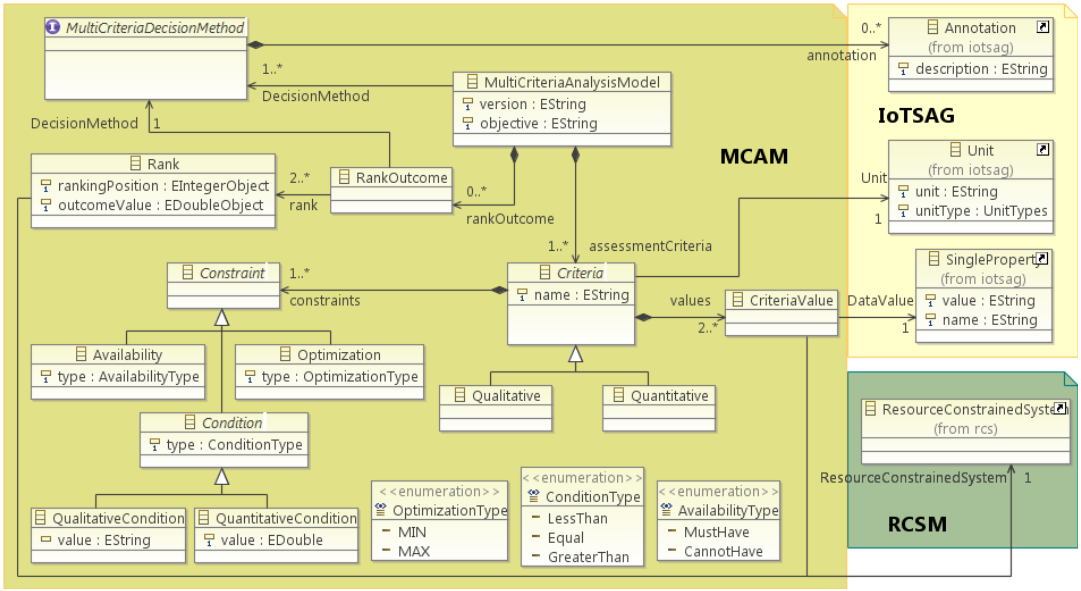


Figure 6.2: Multi-Criteria Analysis Specification Model.

The Ecore representation of MCAM is presented in Appendix D.

A more detail explanation of the MCAM composition is presented, evidencing the use of each defined structure:

- **MultiCriteriaAnalysisModel:** the main model class includes a model version (for version control) currently at "2.0-2019" and a description of the objective. A *MultiCriteriaAnalysisModel* is composed by at least one criterion (class *Criteria*), at least one decision method (class *MultiCriteriaDecisionMethod*) and can contain the outcome result (class *RankOutcome*). With no criterion define it is impossible to deliberate upon the available solutions. With no decision method there is no MCDM methodology to apply, making the proposed methodology incomplete. The outcome result is fulfilled by the IoT Systems Assessment process;

- **Criteria:** is an abstract class to represent two kinds of criteria, *Qualitative* or *Quantitative*. The criterion is defined by its name and has to declare its unit, by using the class *Unit* from the IoTSAG specification. It is also composed by at least one constraint (class *Constraint*) and at least two values for the same criterion (which indicates that there is at least two solutions being evaluated);

- **Qualitative:** is a specific *Criteria* used to define qualitative criteria, for example: implementation difficulty;

- **Quantitative:** is a specific *Criteria* used to define quantitative criteria, such as cost, energy consumption, etc.;

- **CriteriaValue:** this class is composed by a property (*SingleProperty* class) from IoTSAG specification, used to instantiate criterion value, and by the identification of the IoT System (*ResourceConstrainedSystem* class) to which the property value belongs to;

- **Constraint:** is an abstract class to represent the different kinds of criteria constraints. An instantiation of a *Constraint* can be of the following types: *Availability*, *Optimization*, and also of two types of *Condition*, *QualitativeCondition* or *QuantitativeCondition*;

- **Availability:** is a specific *Constraint* used to express the necessity of having a feature within the final solution. A feature's need is defined by the attribute *type* which is of type *AvailabilityType*. The use of this type of constraint can automatically exclude a solution;

- **AvailabilityType:** is an enumeration class. An attribute of this kind can be of two types: *MustHave*, used to indicate that a solution has to include that criterion, and *CannotHave*, used to indicate the opposite, i.e. a criterion cannot be part of the final solution;

- **Optimization:** is a specific *Constraint* used to sort all considered solutions. The attribute *type* defines the sorting type which is of type *OptimizationType*. The use of this type of constraint does not automatically exclude a solution;

- **OptimizationType:** is an enumeration class. An attribute of this kind can take two values: *MIN* where a solution with the less criterion value gains a higher score and *MAX* where a solution with the higher criterion value gains an higher score. Criteria values are ordered in a minimization or maximization sequence;

- **Condition:** is an abstract class to represent two types of conditions: *QualitativeCondition* and *QuantitativeCondition*. It is used to express a data range. The attribute *type* defines the condition type which is of type *ConditionType*. The use of this type of constraint can automatically exclude a solution;

- **QualitativeCondition:** is a specific *Condition*, which inherits *Constraint* features. The attribute value is used in combination with the condition type to set starting point of the data range. For example, the implementation difficulty has to be less than medium difficulty. For qualitative criteria, at this moment, decision makers must indicate the weight for each possible qualitative criterion values (e.g.: Very Important – 10; Important – 5; Not Important – 1). Also, a semantic analysis to the qualitative criterion values could be performed to automatically assign a specific order;

- **QuantitativeCondition:** is a specific *Condition*, which inherits *Constraint* features. The attribute value is used in the same way as for class *QualitativeCondition*, an example could be the cost has to be less than a certain amount of Euros;

- **ConditionType:** is an enumeration class. An attribute of this kind can be of three types: *LessThan*, *Equal*, *GreaterThan*;

- **MultiCriteriaDecisionMethod:** is an interface class that allows instantiation of different MCDM Methods. In this way the proposed framework is not bound to restrict, pre-established decision methods. However, new methods specification models must respect the fact that its main model object has to inherit from the interface class *MultiCriteriaDecisionMethod*. Sections 6.4.1 and 6.4.2 propose specification models for two MCDM Methods, that shows the framework openness to the use of different decision methods;

- **RankOutcome:** is a class than gathers the assessment result. It is composed by the decision method used (class *MultiCriteriaDecisionMethod*) and by at least two ranking positions (class *Rank*), which relates the rank with the possible solutions (IoT Systems);

- **Rank:** identifies, relates the solutions analysed with their outcome. It is composed by the identification of the solution (class *ResourceConstrainedSystem* from the RCS specification model), the solution ranking position and its assessment result value.

The proposed specification model (MCAM) enables stakeholders to assess which IoT System is more suitable to their application scenario and purpose. It is a concrete form to describe criteria than influence the overall performance, define constraints or restrictions for each criterion, and which MCDM method will be applied to assist in the decision. Besides all the known benefits of using MCDM methods (e.g. solve problems with multiple and conflicting criteria, solutions ranking), MCAM gathers in a single formalisation the possibility to complement the decision method with user-defined restrictions. Restrictions that indicate that a solution must have or cannot have a certain feature, define value thresholds (limits) or in which way criteria are organised (minimise or maximise optimization).

With its model-driven nature, MCAM is easy to include, operate with other tools or systems. It provides direct binding with IoT Systems formalisation and also the use of different MCDM methods, even user-defined methods.

6.4 MCDM Methods Specification

Multi-Criteria Decision-Making (MCDM) is a process to decide which is the more suitable solution, from a set of available solutions, by analysis of contradicting criteria [177]. Section 4.2 addressed the most widely used decision making methods in literature in a general form, where three methods were highlighted: AHP, PROMETHEE and ELECTRE. Section 6.1 presented a background research, focus on IoT, regarding in which scenarios decision-making is being applied and which are the methods used. To complement the State-of-the-art, it was proposed a novel framework for a multi-criteria assessment of IoT Systems, with a possibility to define different types of constraints divided into three groups (optimisation by not only by maximisation but also minimisation; availability by must or cannot have a certain feature; and definition of acceptance data ranges by a settling conditions) but also the used of different MCDM methods.

Next is proposed two specification models for the MCDM methods presented in Section 4.2.1 and 4.2.3, respectively the Analytic Hierarchy Process (AHP) and Elimination and Choice Expressing the Reality (ELECTRE). The proposed Meta-Models formalise the MCDM methods' methodologies fulfilling their rules. Each specification model also respects the necessary rule so it can be integrated as one MCDM methods' possibility in the MCAM. This rule refers to the necessity of the main model class to inherit from the interface class *MultiCriteriaDecisionMethod* of the MCAM.

6.4.1 Analytic Hierarchy Process (AHP) Specification

The Figure 6.3 proposes a specification model to describe, formalise the AHP decision method, and to allow its reference, inclusion by the Multi-Criteria Analysis specification model (MCAM). A more detail explication of the AHP specification model, evidencing the use of each structure, is presented next:

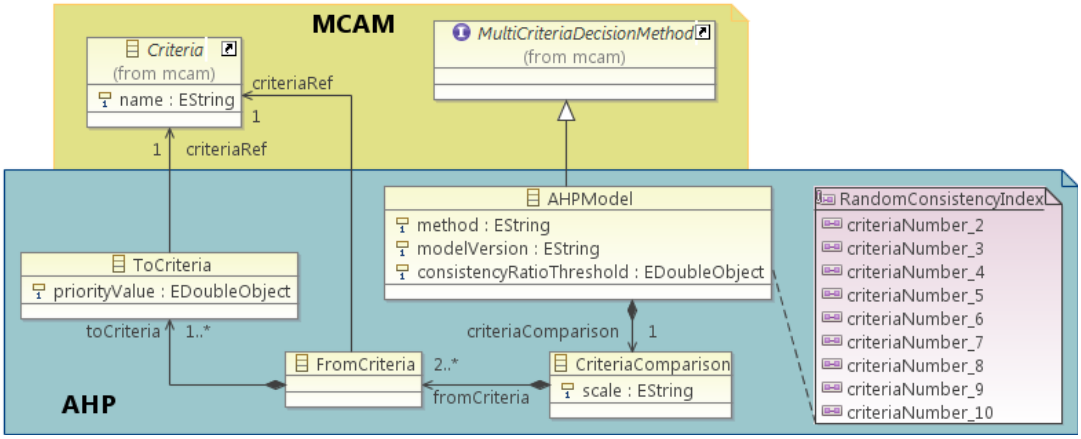


Figure 6.3: Analytic Hierarchy Process (AHP) Meta-Model.

- **AHPModel:** is the main model class and contains three attributes. First, *method*, identifies the MCDM method that in this case is by default "Analytic Hierarchy Process". Second attribute, *modelVersion*, is a model version (for version control) currently at "1.1-2019". The third is the definition of the consistency degree threshold (*consistencyRatioThreshold*), settle by default to 10%, is the value used to analyse correctness of priorities assigned by decision makers. An *AHPModel* is composed by one *CriteriaComparison*, and contains an annotation of type *RandomConsistencyIndex*. It inherits from *MultiCriteriaDecisionMethod* class of the MCAM, there by respecting the rule imposed by the framework;

- **CriteriaComparison:** is a class to aggregate all criteria used by the decision method. Contains one attribute, *scale*, to define the comparison scale used to compare criteria priority. Attribute *scale* is set by default to "Saaty 1-9 scale". This class is composed by at least two criteria (class *FromCriteria*), since it is the minimum number of criteria for this method;

- **FromCriteria:** is a class used to define the pairwise comparison matrix (see Section 4.2.1, Equation 4.4). It is composed by one criterion (reference to *Criteria* class from MCAM), and at least one reference (reference to class *ToCriteria*) to priorities values regarding the other criteria available in the decision process. It is able to define the relation, i.e. priorities values, between criterion *a* and all the other criteria;

- **ToCriteria:** it enables the definition of criteria priority values. Meaning that a source criterion (*FromCriteria*) has a priority value of *priorityValue* attribute in relation to a target criterion, referenced by class *Criteria* from MCAM;

- **RandomConsistencyIndex:** is a specification model annotation that contains nine random consistency index values, from two to ten criteria. The values are the same as in Table 4.1 and were retrieved from [186].

The Ecore representation of AHP specification model is presented in Appendix E.

To give a better notion of the match between the AHP decision process and the proposed AHP specification model, Figure 6.4 depicts the relationship between the Equation 4.4 and the equivalent classes of the proposed AHP Meta-Model. This relationship works in the two directions, meaning that is possible to fill the specification model from the pairwise comparison matrix, AHP_M, as it also possible to build the AHP_M matrix from the AHP Meta-Model.

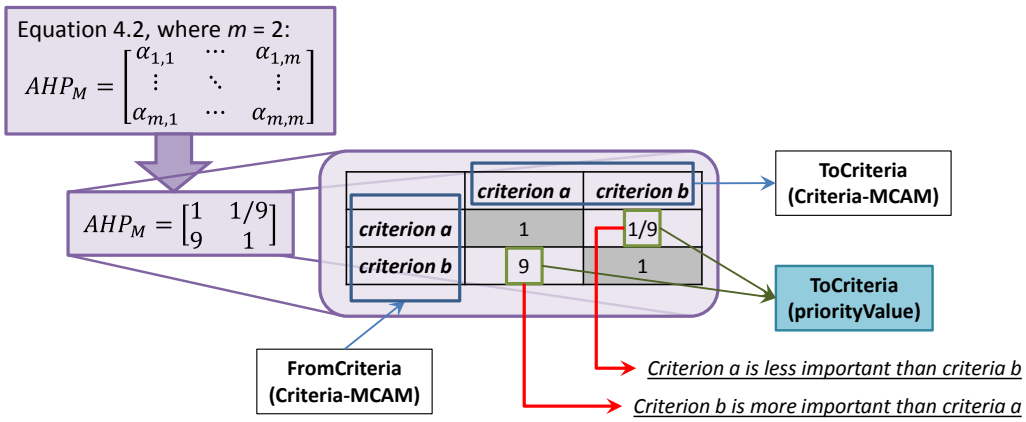


Figure 6.4: AHP Meta-Model: Example of a Pairwise Comparison Matrix.

The example considers two criteria, criterion a and criterion b ($m = 2$). The pairwise comparison matrix, AHP_M, shows that criterion b is nine times more important than criterion a, and consequently criterion a is nine times less important than b. The priority value between the same criterion is always one. The table identifies the specification model classes used.

Furthermore, the decision matrix (Equation 4.3) is criterion value matrix from the solutions versus the criteria considered important. This matrix is obtained from all the instantiations of the *FromCriteria* class. This class references the *Criteria* class from MCAM, that has a connection to each criterion value and the correspondent IoT System to which this value belongs. Equations 4.5 to 4.9 from the decision method AHP, are basically mathematical calculations and consequently no need for a representation in the proposed specification model.

6.4.2 ELECTRE Specification

ELECTRE is one of the decision-making methods point out as a MCDM method widely used in the literature [179–181]. In the same way as for the previous case, next is proposed a specification model for the ELECTRE method, respecting the rules of the ELECTRE method presented in Section 4.2.3 and the proposed framework rule to allow the ELECTRE Meta-Model utilisation in the multi-criteria assessment of IoT Systems.

Figure 6.5 depicts a specification model for ELECTRE decision method. A more detail explanation of the ELECTRE Meta-Model composition is presented, evidencing the use of each defined structure:

- **ELECTREModel:** the core model class, that inherits from the interface class *MultiCriteriaDecisionMethod* (MCAM) to enable its inclusion, use in the proposed Framework for IoT Systems Assessment. This class presents three attributes that identify the model. First is attribute *method* set by default to "ELECTRE", the second designates the model version used for version control (at this point it is set to "1.4-2019"), and the third (*methodVersion*) is used to stipulate which ELECTRE version is specified in the model. An *ELECTREModel* class is composed by the definition of a discordance and one to five concordance threshold levels (class *Coalition*), and by a weight vector (class *CriteriaImportance*) to identify all criteria weights;

- **ELECTRETypes:** is an enumeration class. It is used by the core model class to identify the ELECTRE version. As presented in Section 4.2.3, the ELECTRE method has six variations. The proposed Meta-Model is able to describe five of the six variations, ELECTRE type I, II, III, IV and IS. The specification model is not adequate to describe ELECTRE TRI, since it is not capable to describe the categories assign to the solutions in this ELECTRE variation;

- **Threshold:** is an abstract class to represent the different groups of thresholds. An instantiation of a *Threshold* can be of three types: *Coalition*, *Discriminatory* and *Veto*. A threshold is defined by its value;

- **Coalition:** is a specific *Threshold* used to define the outranking relation between two solutions. The coalition is formed by two conditions, the *Concordance* and *Non-Discordance* conditions, used to define the four possible outranking relations between two solutions (see Section 4.2.3). Variation II and IV of ELECTRE method use two and five concordances levels, respectively;

- **Discriminatory:** is a specific *Threshold* used to define a preference and indifference thresholds to each criterion. *Discriminatory* thresholds are used in ELECTRE III, IV and IS variations in one way or another;

- **Veto:** is a specific *Threshold* used in all ELECTRE variations except in I. ELECTRE Iv is an unofficial variation that refers to ELECTRE I with veto threshold. The veto

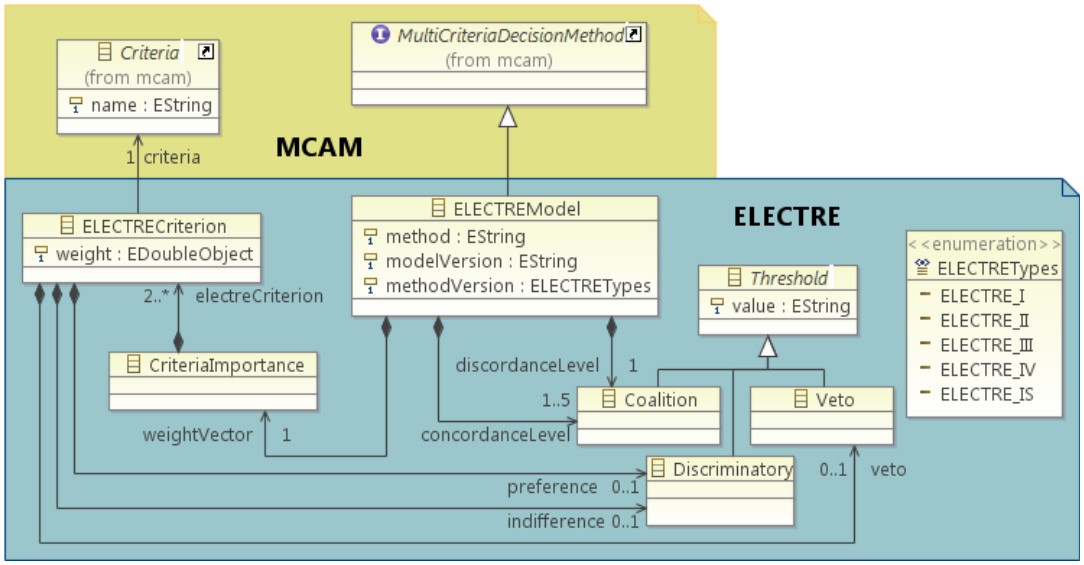


Figure 6.5: Elimination and Choice Expressing the Reality (ELECTRE) Meta-Model.

concept is related to the definition of the discordance level, giving an upper bound to this threshold;

- **CriteriaImportance:** is used to represent the weights assigned to each criterion. This class is composed, aggregates at least two criteria (referencing class *ELECTRE-Criterion*);

- **ELECTRECriterion:** a class that defines all aspects related to criteria which are also directly related with the used ELECTRE method variation. This class is composed by one criterion (reference to Criteria class from MCAM), and it can contain three types of thresholds. Two *Discriminatory* thresholds, identified by *preference* and *indifference* references, and by one *Veto* threshold.

Ecore representation of ELECTRE specification model is presented in Appendix F.

The proposed ELECTRE specification model depicted in Figure 6.5 enables the definition of ELECTRE I, II, III, IV (electre four) and IS variations, being also possible to specify the "unofficial" ELECTRE Iv (electre one vee) variation (basically, variation I with veto threshold definition). However, and given that the specification of MCDM methods is not a focus of this thesis, ELECTRE TRI variation is not contemplated in the proposed Meta-Model. ELECTRE TRI variation focuses on assigning categories to solutions, sorting from the worst to the best category (or in the other way around). The specific feature, assigning categories, that the proposed Meta-Model does not cover.

6.5 IoT Systems: Multi-Criteria Assessment Methodology

The previous sections addressed and formalised, following a model-driven approach, the assessment framework which supports a multi-criteria assessment process for IoT

Systems. Specification models were present that in conjunction with the proposed IoT System specification in Chapter 5, enables stakeholders to describe all factors needed to assess which IoT System is more suitable for a concrete application. These specification models are not only qualified to evaluate and rank solutions based on criteria using known MCDM methods, but also able to use new or even user-defined decision methods. Furthermore, and to tackle the inadequacy of the MCDM methods to define specific constraints/restrictions, it is enabled the specification, association of acceptable/not acceptable data ranges, establishing availability degrees, and optimisation rules for the criteria values.

Although, the proposed framework and each of its steps are well specified, which facilitates the assessment process, it does not implement it. Based on formal descriptions, tools and systems are easily developed, adapt and respond well to changes [147].

Consequently, next is presented the IoT Systems Multi-Criteria Assessment Methodology, based on the proposed specification models and within the Framework depicted in Figure 6.1, more effectively in the *Assessment Engine* block, to actually implement, materialise the assessment of IoT Systems. To manage the model-driven nature of the proposed framework, support the harmonisation between meta-models, models and data it is used the MDA-based Harmonisation Framework presented in Section 5.5, more precisely in Figure 5.14.

The proposed methodology starts by collecting the available IoT Systems, or by gathering the ones which will be analysed. As mentioned in Chapter 5 users can also build new IoT Systems based on different parts (e.g.: micro-controllers, communication boards, sensing elements, etc.). From the formal descriptions, solutions, i.e. IoT Systems, can be selected, identify by stakeholders (users, developers, researchers, etc.) creating a solutions group to be analyse. These ***Solutions*** are represented by set S_{Set} in Equation 4.1. The number of alternatives, solutions is a finite number $n \in \mathbb{N}$.

The selection of criteria, features is performed following the same principle as in solutions. Criteria are also well described by the proposed specification models (in particularly by IoTSAG *Property* class), making the data clear and accessible for stakeholders. IoT Systems have a vast number of features, although stakeholders may or may not consider all of them as important for the scenario, problem in hand. The set of ***features/criteria***, called C_{Set}, is defined as in Equation 4.2, with a m size, $m \in \mathbb{N}$.

From the MCAM specification (see Figure 6.2) it is possible to build the ***Solutions*** and ***Criteria*** sets, S_{Set} and C_{Set}, respectively. Each instantiation of class *Criteria* is a criterion c_j, i.e. an element of C_{Set}. In the solutions case, the instantiation of class *ResourceConstraintSystem* gives origin to ***Solutions*** set S_{Set}. A solution s_i, with $s_i \in S_{Set}$, is an instantiation of class *ResourceConstraintSystem*, where there is no elements repetition.

With the sets of possible solutions and criteria defined is then possible to create an Assessment Table. The idea of table follows a common principle seen in MCDM methods analysed, and therefore applied to describe the solutions-criteria cluster in the form of a matrix (which also benefits the mathematical operations). The ***Assessment Table***, A_{table},

given by Equation 6.1, is a ***n***-by-***m*** matrix. Rows represent the contemplated solutions and the columns the assessment features. The element $v_{i,j}$ presents the value for IoT System ***i*** of the criterion ***j***. This matrix is in all identical to *DM* matrix presented in Equation 4.3.

$$A_{table} = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,m} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & v_{n,m} \end{bmatrix} = (v_{i,j}) \in \mathbb{R}^{n \times m} \tag{6.1}$$

The possible solutions, IoT Systems, present a vast number of criteria, and each criterion has its own unit. At this point is important to remark that in the proposed formal descriptions, criteria are specified by the IoTSAG *Property* class which is composed by a unit and property domain. Two criterion/property descriptions that work as foundation to outline a method, a process that unifies all units found for the same criterion. Clarification regarding how units and properties domains are treated in the proposed frameworks was addressed in Section 5.3 with a practical depicted in Figure 5.4. Solved the issue of different unit types for the same criterion, the methodology manages all features/criteria values as dimensionless.

As stated, the proposed Multi-Criteria Assessment Methodology for IoT Systems complements the available MCDM methods by enabling the definition of specific constraints/restrictions for each criterion. Seven different constraints are included in the Multi-Criteria Assessment specification model (MCAM), and each one can be applied to one or more criteria, as well as two or more constraints can be applied to the same criterion.

The ***Assessment Constraints*** set, AC_{set}, defined in Equation 6.5, is formed by the three subsets of constraints. This division occurs due to two factors. First the eliminative nature of constraints (automatically exclude solutions) of type *Availability* and *Condition*. Second, *Optimization* constraints have a direct influence on the MCDM methods, since these are built to rank solutions from criteria higher values to the lower ones, and which is why the two constraints subsets are applied at different steps of the proposed methodology (as explained later). In this sense, the ***no-eliminative*** **assessment constraints** set, AC_{Opt}, is defined in Equation 6.2, and the ***eliminative*** **assessment constraints** sets, AC_{Cod} and AC_{Ava}, is defined in Equation 6.3 and 6.4, respectively.

$$AC_{Opt} = \{MIN, MAX\} \tag{6.2}$$

$$AC_{Cod} = \{LessThan, Equal, GreaterThan\} \tag{6.3}$$

$$AC_{Ava} = \{MustHave, CannotHave\} \tag{6.4}$$

$$AC_{set} = AC_{Opt} \cup AC_{Cod} \cup AC_{Ava} \tag{6.5}$$

The stakeholder's judgement has an important role since it is their considerations that specify the rules over the criteria (settling the constraints) which express formally what has to be analysed and in which way, to achieve the objective. The constraints that each criterion has to respect are obtained from the instantiation of MCAM, enabling the proposed methodology to assess all criteria accordingly to decision maker objective.

To determine the impact that the constraints enforce in the solutions, a procedure was developed to execute this task and it is presented in Figure 6.6. This function, ***ProcessAC***, which processes the assessment constraints has four input arguments and returns a value of type double. The argument ***CriteriaValue*** is $v_{i,j}$, the ***ConstraintType*** is one of the AC_{set} elements, the ***MaxValue*** is determine by Equation 6.6, and the last argument ***Threshold*** is the value define in the MCAM for the constraint *Condition* of criterion ***j***.

$$MaxValue = max\{v_{1,j}, v_{2,j}, ..., v_{n,j}\} \tag{6.6}$$

```
Function ProcessAC (CriteriaValue, ConstraintType, MaxValue, Threshold)
  If ConstraintType is of type MAX then
    Return CriteriaValue;
  Else if ConstraintType is of type MIN then
    Return MaxValue - CriteriaValue;
  Else if ConstraintType is of type MustHave then
    If CriteriaValue different from 0 then
      Return 1.0;
  Else if ConstraintType is of type CannotHave then
    If CriteriaValue equal to 0 then
      Return 1.0;
  Else if ConstraintType is of type LessThan then
    If CriteriaValue is less than Threshold then
      Return 1.0;
  Else if ConstraintType is of type GreatThan then
    If CriteriaValue is greater than Threshold then
      Return 1.0;
  Else if ConstraintType is of type Equal then
    If CriteriaValue is equal to Threshold
      Return 1.0;
  Return 0.0;
End Function.
```

Figure 6.6: Procedure to Process Assessment Constraints.

As mentioned, constraints are divided in two sets, *no-eliminative* and *eliminative*, and are applied at different stages of the IoT Systems assessment methodology. The first

assessment constraints applied are the ***no-eliminative evaluation constraints***, AC_{Opt}. Equation 6.7 presents the application of *Optimization* constraints to the solutions-criteria cluster using the Assessment Constraints function, resulting in $OptA_{table}$, a $\boldsymbol{n}$-by-$\boldsymbol{m}$ matrix. Rows continue to represent the contemplated solutions and the columns the assessment features, similar to A_{table} in Equation 6.1. The value $\boldsymbol{p}$ identifies the number of constraints applied to criterion $\boldsymbol{j}$.

$$\begin{gathered}\Phi_{i,j} = \prod_{k=1}^{p} ProcessAC(v_{i,j}, Constraint_k, MaxValue, 0);\\ OptA_{table} = (\Phi_{i,j});\\ Constraint_k \in AC_{Opt};\\ p \in \mathbb{N}\end{gathered} \tag{6.7}$$

The main difference between matrix A_{table} and $OptA_{table}$ is that the second contemplates the application of *Optimization* constraints, which is highly important if stakeholders desire any criterion minimisation analysis.

The proposed framework enables the possibility to apply different MCDM methods, even new or user-defined methods. This MCDM methods diversity makes it impossible to describe all methods in a single, unique mathematical procedure, since it will depend directly of the method used. However, it is known what is needed as input to apply a MCDM method and it is possible to specify how the outcome, result form should be, so the developed enablers can be built to achieve this goal. Therefore, this procedure depicted as a "black box" in Figure 6.7, is basically a process that computes, applies the selected MCDM method.

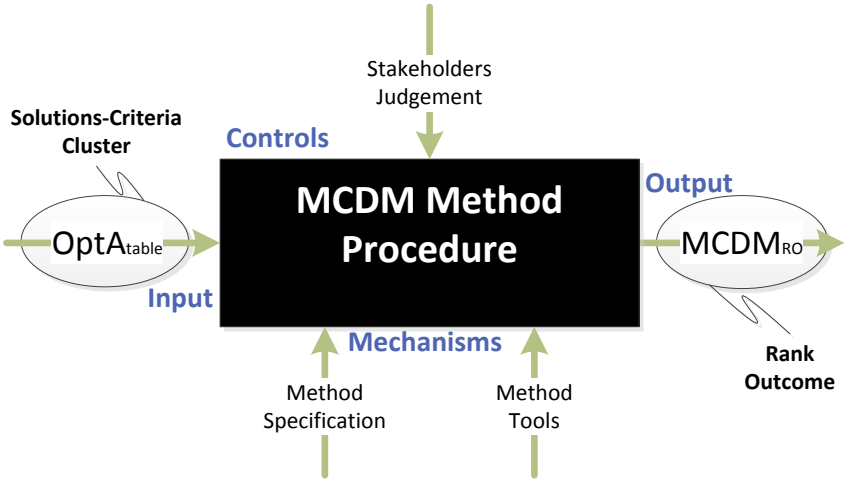


Figure 6.7: MCDM Method Procedure Activity Detail.

The process is presented using a high-level IDEF-0 [196] "black box" view to describe the MCDM method procedure activity specifying inputs, outputs, control, and mechanisms:

- **Input:** the Assessment Table with Optimisation constraints, $OptA_{table}$, i.e. the solutions-criteria cluster with optimisation constraints already applied. Literature review preformed in Section 4.2 show that MCDM methods for problems with a finite and known number of alternatives perform their analysis based on three aspects: information regarding solutions and their features, and in stakeholders judgement. Stakeholder's judgement is part of the control definition, that with the application of the indicated mechanisms the output is calculated;

- **Controls:** the stakeholder's judgement. MCDM methods in some way need the decision maker judgement. For example in the AHP method it is used to establish criterion priority, in PROMETHEE case is used to select the preference functions, and in ELECTRE is used to select the different thresholds. With a model-based framework the stakeholder's judgement is provided in the form of a model (e.g.: AHP model — an instantiation of AHP Meta-Model);

- **Mechanisms:** normally it is the who or what that executes the process. With the information provided by the input and definitions by the controls, it is necessary to apply tools to compute the method outcome. Since the proposed framework as a model-driven nature, specification model is provided describing the method;

- **Output:** the system output is the rank outcome, $MCDM_{RO}$, a result from the application of MCDM method procedure.

The proposed methodology specifies the outcome result provided by the selected MCDM method, ***MCDM Rank Outcome***, as given in Equation 6.8. The ***MCDM Rank Outcome***, $MCDM_{RO}$, is a $\boldsymbol{n}$-by-$\boldsymbol{1}$ matrix. Rows continue to represent each contemplated solution and the column reports the rank outcome values. The element $RO_{i,1}$ presents the MCDM method rank outcome value for IoT System $\boldsymbol{i}$.

$$MCDM_{RO} = \begin{bmatrix} RO_1 \\ RO_2 \\ \vdots \\ RO_n \end{bmatrix} = (RO_{i,1}) \in \mathbb{R}^{n \times 1} \tag{6.8}$$

Known the result from the selected MCDM method is time to apply the two types of assessment constraints left, the ***eliminative assessment constraints***, AC_{Cod} and AC_{Ava}. The enforcement of assessment constraints of type *Condition*, AC_{Cod}, results in a $\boldsymbol{n}$-by-$\boldsymbol{1}$ binary matrix, $Eval_{Cod}$, given by Equation 6.9. The $\boldsymbol{Th}$ argument is the threshold value for constraint $\boldsymbol{k}$ in criterion $\boldsymbol{j}$, value that is retrieved from the instantiation of MCAM.

Considering,

$$\Theta_i = \prod_{j=1}^{m} \sum_{k=1}^{p} ProcessAC(v_{i,j}, Constraint_k, 0, Th);$$
$$Constraint_k \in AC_{Cod};$$
$$p, \in \mathbb{N};$$

then,

$$(Eval_{Cod})_{i,1} = \begin{cases} 1, & \textbf{if } \Theta_i > 0 \\ 0, & \textbf{otherwise}. \end{cases} \tag{6.9}$$

The enforcement of the ***assessment constraints*** of type *Availability*, AC_{Ava}, also results in a ***n***-by-***1*** binary matrix, $Eval_{Ava}$, and is given by Equation 6.10.

$$\begin{aligned} (Eval_{Ava})_{i,1} = \prod_{j=1}^{m} \sum_{k=1}^{p} ProcessAC(v_{i,j}, Constraint_k, 0, 0); \\ Constraint_k \in AC_{Ava}; \\ p, \in \mathbb{N} \end{aligned} \tag{6.10}$$

The binary outcome from matrices $Eval_{Cod}$ and $Eval_{Ava}$ state if a solution is considered valid according to stakeholders' *eliminative* constraints. The ***final solutions ranking***, $(IoT_{Systems})_{Rank}$, is given by Equation 6.11. Where $(S_{Value}^{Outcome})_{i,1}$ is the Multi-Criteria Assessment outcome value for the solution ***i*** (IoT System ***i***). The highest $(S_{Value}^{Outcome})_{i,1}$ identifies the more suitable solution, according to the proposed methodology.

$$\begin{aligned} (IoT_{Systems})_{Rank} &= MCDM_{RO} \odot Eval_{Cod} \odot Eval_{Ava} \\ &= \begin{bmatrix} (S_{Value}^{Outcome})_1 \\ (S_{Value}^{Outcome})_2 \\ \vdots \\ (S_{Value}^{Outcome})_n \end{bmatrix} = ((S_{Value}^{Outcome})_{i,1}) \in \mathbb{R}^{n \times 1} \end{aligned} \tag{6.11}$$

6.6 Topic Discussion

This Chapter addressed a multi-criteria decision problem regarding the more suitable IoT System to perform a certain task. New embedded systems are provided everyday by manufacturers, and many are the features/criteria that influence the overall performance of an IoT System, which makes the IoT Systems selection a decision very difficult for stakeholders (e.g. engineers, developers, end-users, etc.) [32, 203].

It was proposed a novel framework and a multi-criteria specification model to analyse IoT Systems, covering hardware, software as well as energy consumption aspects. Presents the capability to allow integration of different MCDM methods (even new ones) to analyse and rank solutions, but it also introduces means for stakeholders to define different types of criteria constraints. Decision makers can indicate optimisation functions

(best or worse based on criteria value), availability restrictions (must or cannot have a certain feature) or set data range for criteria for which a solution is acceptable.

In this sense, it is addressed the question, Q1.2, raised in Section 1.4.1, stating : "*Which methods could be applied or develop to assist in IoT System assessment?*". Not only, the full State-of-the-Art regarding Multi-Criteria Decision-Making (MCDM) for problem with a known number of solutions is considered, it is suitable to be used within the proposed framework. But also, new tools are provided to work together with MCDM methods, improving the assertiveness of the decision.

Furthermore, as secondary contribution are proposed specification models for two well-known MCDM methods in literature. The Meta-Model are for the Analytic Hierarchy Process (AHP) and for Elimination and Choice Expressing the Reality (ELECTRE) methods. With the remark, that the proposed ELECTRE specification model is not able to describe the ELECTRE TRI method variation (the sixth method variation).

Disadvantages from the known MCDM methods are not solve and are projected to the proposed methodology, however the framework openness to accept new, different MCDM methods allows stakeholders to decide which is best not only for them but also for each particular application case. Besides, with the model-driven nature of the framework, already existing MCDM methods can be improved, and changes applied. It is possible, by updating a MCDM method specification model and the respective interoperability enablers, to continue using the proposed framework.

It is also important to notice that is possible to specify qualitative criteria. Decision makers can indicate a weight for each possible qualitative criterion value (e.g.: Very Important – 10; Important – 5; Not Important – 1), or a semantic analysis could be made to automatically assign quantitative values for the qualitative criterion. The study or applicability of semantic analysis is not one of the focuses of this thesis.

Chapter 7

Framework for Design Support of IoT Systems

This chapter intends to present the author's main conceptual contribution, an IoT Systems design support framework based on IoT Systems detailed and formal characterisation and in a multi-criteria assessment methodology for IoT Systems. Stakeholders are provided with an effective toolset to make decisions reasoning, more aware during IoT Systems design phase [32]. Coexistence with other tools, systems, standards or methods is also taken into consideration by the model-driven nature of the proposed framework and well-defined formalisations [204]. An example is the benefit that energy consumption evaluation tools can obtain from the proposed IoT Systems formal specifications. Energy usage optimization depends on modelling power consumption. Model-based simulation must consider parameters that depend on the device used, operating system, and application under study [195].

7.1 Conceptual Approach for Design Support of IoT Systems

The author's conceptual approach for the design support of IoT Systems is based on two points identified from the background research. The first focus on lack of methods to describe and assess the suitability of IoT Systems for a specific task and the second on the mechanisms considered ideal to assist on achieving the desired goal. This goes in line with the main research question formalisation in Section 1.4 stating: *"How can Internet-of-Things Systems be designed to optimize the matching with the operating environment?"*

Figure 7.1 depicts the author's main conceptual contribution overview, for the *Multi-Criteria Framework to Assist on the Design of IoT Systems*. The conceptual approach initial step, ***Alternatives***, is in all related with the wide diversity of "things" (IoT Systems)

provided by manufacturers, designed for different purposes, addressing a variety of application domains and services. Consequently, the set of alternatives must be collected from webpages, systems, modelling tools, datasheets, etc., enabling IoT Systems definitions (formal specifications) in terms of hardware, software and other general information considered important. IoT Systems formal specification is accomplished with the formalisations proposed in Chapter 5.

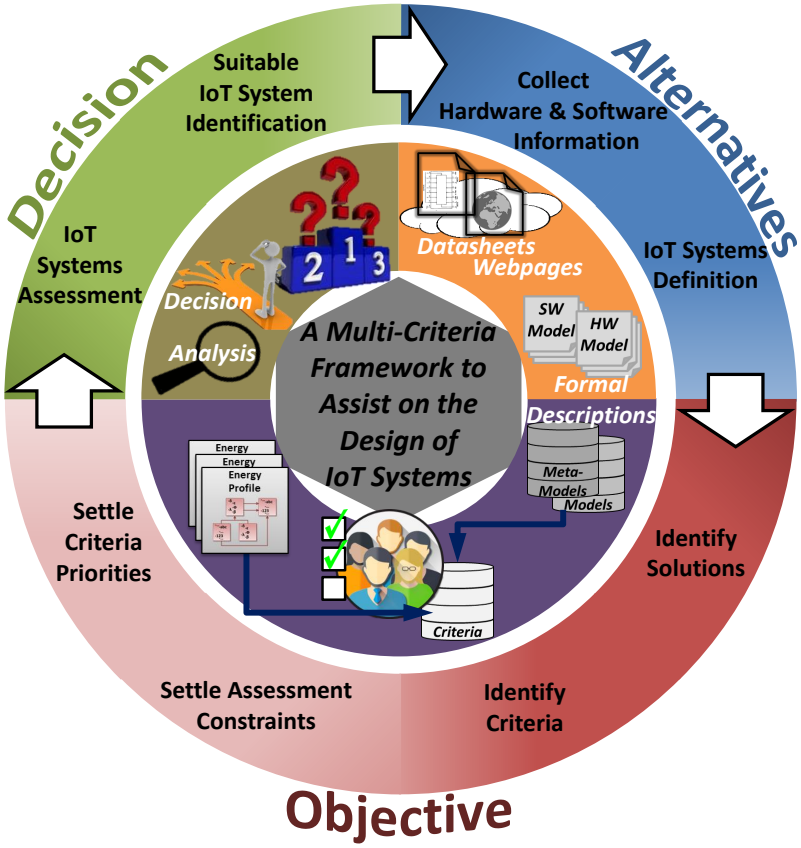


Figure 7.1: Proposed Concept for the Design Support of IoT Systems.

The second step, ***Objective***, is related to the task, problem to solve definition. It starts by identify, select possible solutions (IoT Systems) from the set of alternatives specified in the previous step. Although it is possible to consider the entire solutions set, database, the stakeholders may decide for some reason that there are some IoT Systems that should not be considered as possibilities for a specific task. IoT Systems are mainly used, but not limited to, for environment awareness, giving people a more truly interaction with the surrounding world. Sensing the environment is an operation that can change from case to case. Different sensor units imply different hardware features, or transmission data sets, or communication protocols, etc. Consequently, for each case the task objective(s) must be defined. To accomplish this, is envisage the criteria identification (e.g.: sensing unit type, deployment difficulty, transmission range, cost, etc.), specify criteria constraints (e.g.: sensor of humidity type, deployment difficulty less or equal to medium, transmission above 10 meters, etc.) and define which are the more important criteria (normally criteria

does not have the same importance for stakeholders, and this can change from case to case) for the established objective.

The third and final step, ***Decision***, addresses the decision regarding the suitable solution(s) for a specific task. Many are the factors which influence the overall performance of an application running in an IoT System. Combine or fulfil several requirements is not an easy task. Therefore, to perform a more conscious decision of which is the proper solution(s), is applied one of the most widely used decision methodologies — Multi-Criteria Decision-Making (MCDM). IoT Systems multi-criteria assessment is accomplished by combining MCDM methods with a constraint-based assessment methodology. Constraints are set accordingly with three principles: optimisation rules; acceptable or unacceptable data ranges; and mandatory or non-mandatory criteria availability. Matching decision-makers prerequisites imposed during the design of an IoT System, is possible to identify the more suitable solution (IoT System) through the decision process outcome.

The author's approach aims to provide a mechanism to assist stakeholders on the design of IoT Systems, enabling conscious and justifiable decisions upon more suitable IoT System(s), strengthened with modules to accomplish interoperability with systems and tools (e.g. Standards, integration with energy-assessment tools).

7.2 Framework for Design Support of IoT Systems

To materialise the high level abstraction structure presented in Figure 7.1, is proposed a framework for the design support of IoT Systems in Figure 7.2. The proposed framework also addresses interoperability and harmonisation with other tools, systems, standards or methods. Highlighting, the possible positive impact on energy consumption simulation tools (addressed in Section 5.4.3 — ***IoT System: Energy Profile Formalisation***) and highly interoperable nature (that will be addressed in Section 8.1.2 — ***Application Scenario 2: SensorML Standard***, and demonstrated in Section 8.2.2 — ***Implementation of Scenario 2: SensorML Standard***).

The Multi-Criteria Framework for Design Support of IoT Systems presented in Figure 7.2, aggregates the two previous contributions, presented in Chapter 5 — ***Framework to Formally Describe an IoT System***, and in Chapter 6 — ***Assessment Framework for IoT Systems***.

The formal specification of IoT Systems is identified in the figure by **IoT System** block, and assessment of IoT Systems is identified by **Assessment Methodology** block. As described in the framework presented in Section 5.2 an IoT System is characterised at two levels. At a higher level (Platform-Independent, abstraction level 2 as presented by MDA) the formal specification (a generic IoT System) and at a lower level (Platform-Specific, abstraction level 1 as stated by MDA) the IoT System description accordingly, respecting the formal specifications (a specific IoT System).

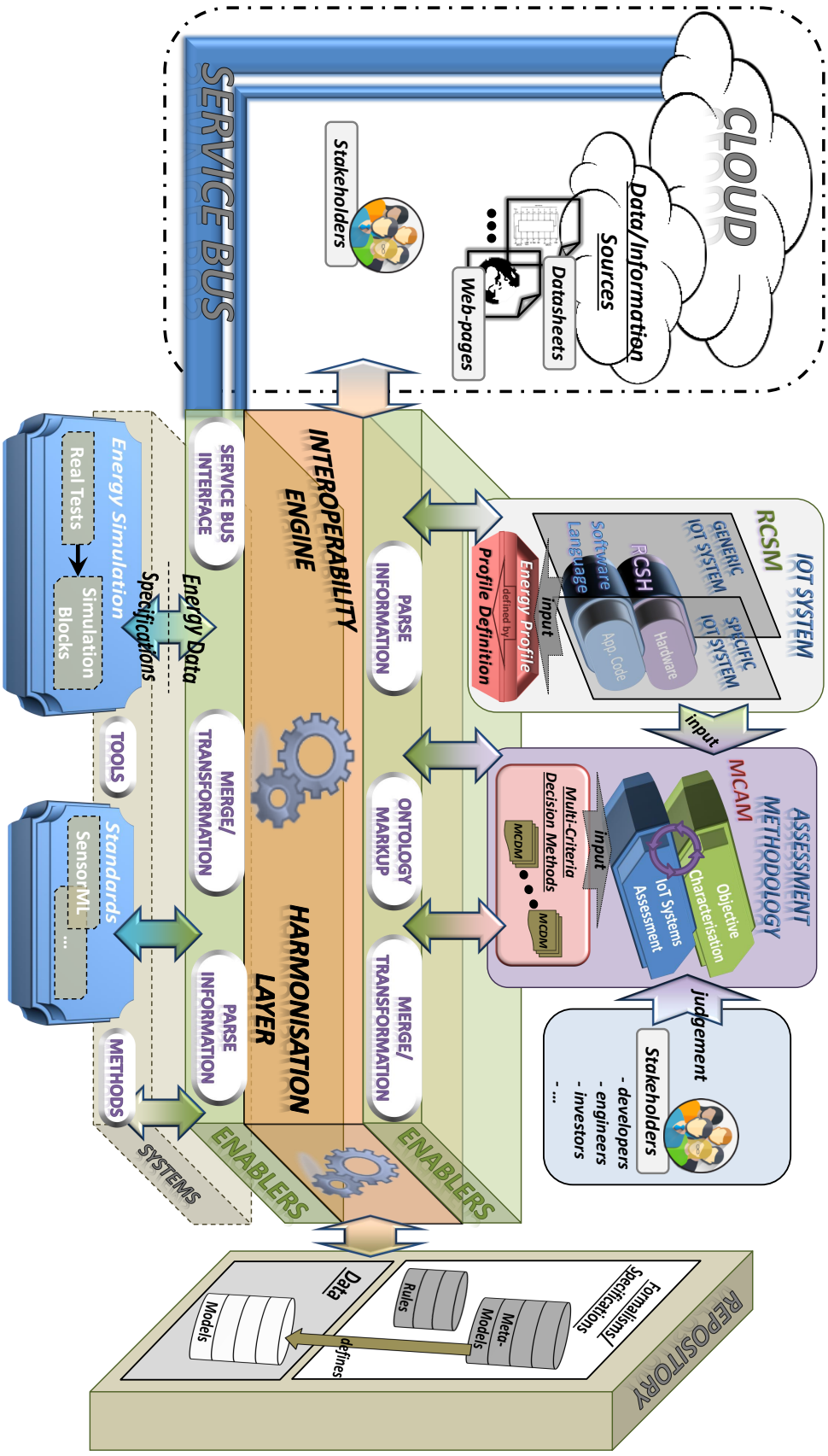


Figure 7.2: Multi-Criteria Framework for Design Support of IoT Systems.

The assessment of IoT Systems is established based on two principles: 1) a formal characterisation of the objectives using criteria, different and predefined constraints types, and taking into consideration stakeholders preferences; 2) an IoT Systems assessment methodology to analyse, enforce all conditions that define the objective, and with the capability to apply any MCDM method(s) to rank all solutions.

These two blocks, **IoT System** and **Assessment Methodology**, result of the proposed contributions in Chapter 5 and 6, will not be addressed in detail here since it was already done in the previous chapters.

The Multi-Criteria Framework for Design Support of IoT Systems (see Figure 7.2), presents a core block that blends, conciliate, works to a specific end — make all other blocks and components to work as a whole. This block is the ***Harmonisation Layer***.

The ***Harmonisation Layer*** takes use of an ***Interoperability Engine*** to execute and manage a set of ***Enablers***. These ***Enablers*** are framework components built to interpret, orchestrate distinct data sets (structured or not) from different sources, capable of import/export data enabling the proper framework functioning as well as maintain, sustain interoperability with other systems or tools. Examples of structured data sets are standard specification models or the very specifications of IoT Systems presented earlier in this work. Unstructured data are for example the description of IoT Systems hardware components available in manufacturer's web-pages. The ***Interoperability Engine*** is a processing engine responsible to execute already available ***Enablers*** or include new ones. Inclusion of ***Enablers*** occurs whenever a new specification model, system, tools, etc. is considered to work/integrate the proposed framework. Engineers, developers have to provide, always, the respective enabler. The ***Interoperability Engine*** is based on MDA techniques as proposed by Agostinho et al. [197] and on the author work developed in [198], which addresses interoperability problems associated to model languages transformations, and furthermore allowing communities to build interoperable systems and services. ***Harmonisation Layer*** and ***Interoperability Engine*** will be address later in Section 7.4.

Another framework block is ***Repository***, an "aggregation" of the ***Storage*** blocks presented in the two previous frameworks (see Section 5.2 and 6.2), that follows the same approach — a two levels block. Top level, ***Formalisms/Specifications***, to store formalisation, specification models (meta-models) for standards, tools, methods or systems and respective mapping rules that regulate how data interchange is performed between models. Bottom level, ***Data***, is a storage unit for data models that are created and consequently have to be stored along the framework functional process.

Illustrated at the bottom of Figure 7.2 is the ***Systems*** block. The word "systems" is used in this block in a general, broad sense, referring to a block that embraces a wide set of unknown possibilities (e.g.: tools, methods, standards, systems, etc.). Section 6.4 proposes specification models for two different MCDM methods to be used within the framework by the assessment methodology which follows under this category. Particularly, it is highlighted two "systems", ***Energy Simulation*** and ***Standards***.

Energy Simulation is of great relevance within IoT scope [7–9, 11, 33, 39, 40], and directly related with the third point to describe an IoT System — ***Energy Profile***. Section 5.4.3 addressed this matter by proposing IDEF-0 [196] "black box" view on an energy analysis activity indicating that a trade-off can be made between the proposed IoT Systems specifications and simulation results of energy consumption. It is indicated, and depicted in the figure, that formal, specific descriptions of IoT Systems can be applied, used by simulations tools with the return of energy data to create an ***Energy Profile***.

Another point is interoperability with ***Standards***. This work proposes means to assist during the design phase of IoT Systems, however other phases exist like deployment or observation. As mentioned in Section 5.1, literature has been more focus on functional/behaviour aspects, and in activities/actions/interactions within an IoT Deployment [16–18]. Proof of this is the creation of standards such as the Systems Modeling Language (SysML) [19, 20], the Sensor Model Language (SensorML) [21] and Semantic Sensor Network (SSN) [22]. As an additional feature to the already presented contributions is proposed, based on the model-driven framework nature, means to interconnect the results of the design phase (identification of the more suitable IoT System(s), with well-formed descriptions) with the following phases of an IoT Ecosystem build up. This aspect will be addressed in Sections 8.1.2 and 8.2.2, presenting an accomplished interoperability with SensorML standard, therefore highlighted in the proposed Multi-Criteria Framework for Design Support of IoT Systems, depicted in Figure 7.2.

7.3 Design of IoT Systems: Specification Models

Chapters 5 and 6 addressed two core contributions of this work: a *Framework to Formally Describe an IoT System* and an *Assessment Framework for IoT Systems*, respectively. Despite the clear relation between these contributions, it is still necessary to formally present the connection specification, the glue that binds, aggregates both of them. Furthermore, when designing a System of IoT Systems is needed to contemplate a group of IoT Systems and with the possibility, almost certain, of each IoT System have its own objective, task in the overall system.

Figure 7.3 proposes the IoT Systems Assessment Core (IoTSAC) specification model. Through this Meta-Model is possible to define/reference known RCS (possible solutions), specify different objectives, criteria and constrains (definition of tasks to be performed), and select/apply different MCDM methods, even for the same goal. Consequently, a complete formalisation is provided, specifying how to perform a more aware, proper decision regarding the selection of an IoT System for distinct tasks that needs to be executed in an entire system of IoT Systems. A more detailed explanation of the IoTSAC Meta-Model composition is presented next, evidencing the use of the two defined structures:

- **TheSystemIoT:** is the main model class. It inherits all features from IoTSAG *SystemDescription* class, used to register, give identification to the entire system of IoT

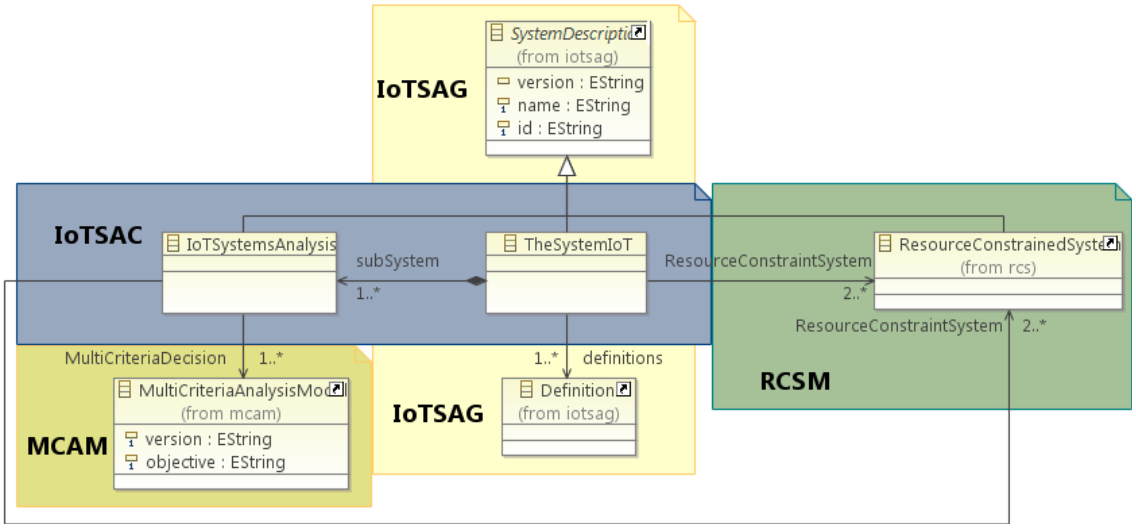


Figure 7.3: IoT Systems Assessment Core Specification Model.

Systems, i.e. the IoT scenario, deployment. As mentioned, a system of IoT Systems can be, and normally is, built upon a set of different tasks, functions which are performed by autonomous sensors. Each task, function has its own requisites and objective(s). The analysis of the considered IoT Systems to perform such task is represented by the class *IoTSystemsAnalysis*. The *TheSystemIoT* class identifies all RCS considered valid for evaluation, being used or not, by a reference to the respective RCS models (*ResourceConstraintSystem* link). The IoTSAC specification model is also responsible to identify all definitions used, from one or several IoTSAG models. Once identified the IoT Systems for assessment, the proposed framework is responsible for managing properties definitions, for example verify, aggregate units and property domains;

- **IoTSystemsAnalysis:** a class used to combine all IoT Systems considered valid and describe the applied multi-criteria assessment, that ascertains which is/are the more suitable IoT Systems for a certain objective. RCS considered valid are reference using the link to *ResourceConstrainedSystem* class from RCSM. Note that for each assessment makes sense consider at least two IoT Systems for comparison. Different multi-criteria assessments can be set to evaluate IoT Systems for the same objective, for example by changing a constraint. Class *IoTSystemsAnalysis* also inherits all features from IoTSAG *SystemDescription* class, used to register and give identification to each IoT Systems assessment scenario.

The Ecore representation of IoTSAC Meta-Model is presented in Appendix G.

The relationship between all specifications models, identified as packages, which formalise the necessary aspects within the proposed framework are depicted in Figure 7.4, with a high level view of the involved packages structure. In a five permanent packages disposition, others exist to describe non-static, changeable aspects of the framework.

Static packages are the IoTSAG, IoTSAC, RCSM, RCSH, and MCAM. Non-Static, changeable packages refer to the ability to include different MCDM methods, software languages and energy consumption profiles. In the figure are used two types of relation (arrows): *uses* and *imports*. The relation *uses* intends to demonstrate that one or more classes within a package reference a class(es) in another package. Relation *imports* demonstrate that one or more classes within a package inherit from a class of another package. This relation can also include a *uses* relation type.

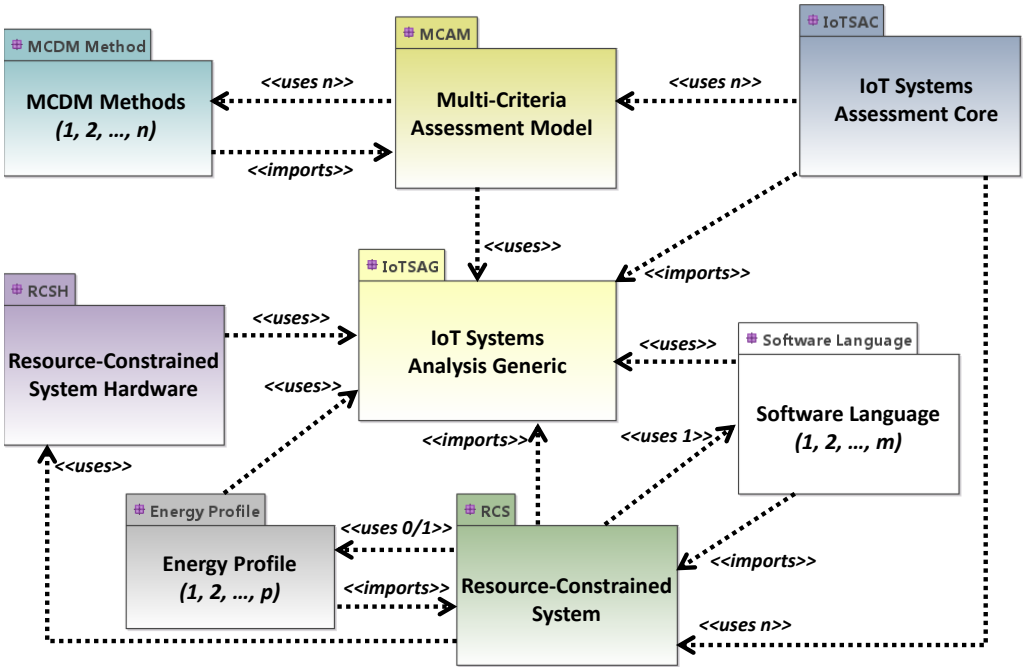


Figure 7.4: Design Support of IoT Systems: High Level Packages Structure.

Accordingly, the core specification (IoTSAC) references one or more IoT Systems (RCSM), one or more multi-criteria assessment (MCAM), and includes (and references) features from IoTSAG (generic, common aspects specification model). A multi-criteria assessment description references one or more MCDM methods and a generic aspects description (IoTSAG). Each MCDM method specification inherits from MCAM to respect integration rules, as described in Section 6.4. IoT Systems (RCSM) definitions inherit description features and references properties from IoTSAG. To its full representation is reference also a hardware description (RCSH), a software language and it can also include an energy profile description. These last two have to respect the framework integration rules. The RCSH, chosen software language and if it is the case an energy profile description reference properties from IoTSAG specification.

Figure 7.5 presents in detail the relationship between the five main specification models, and their respective classes.

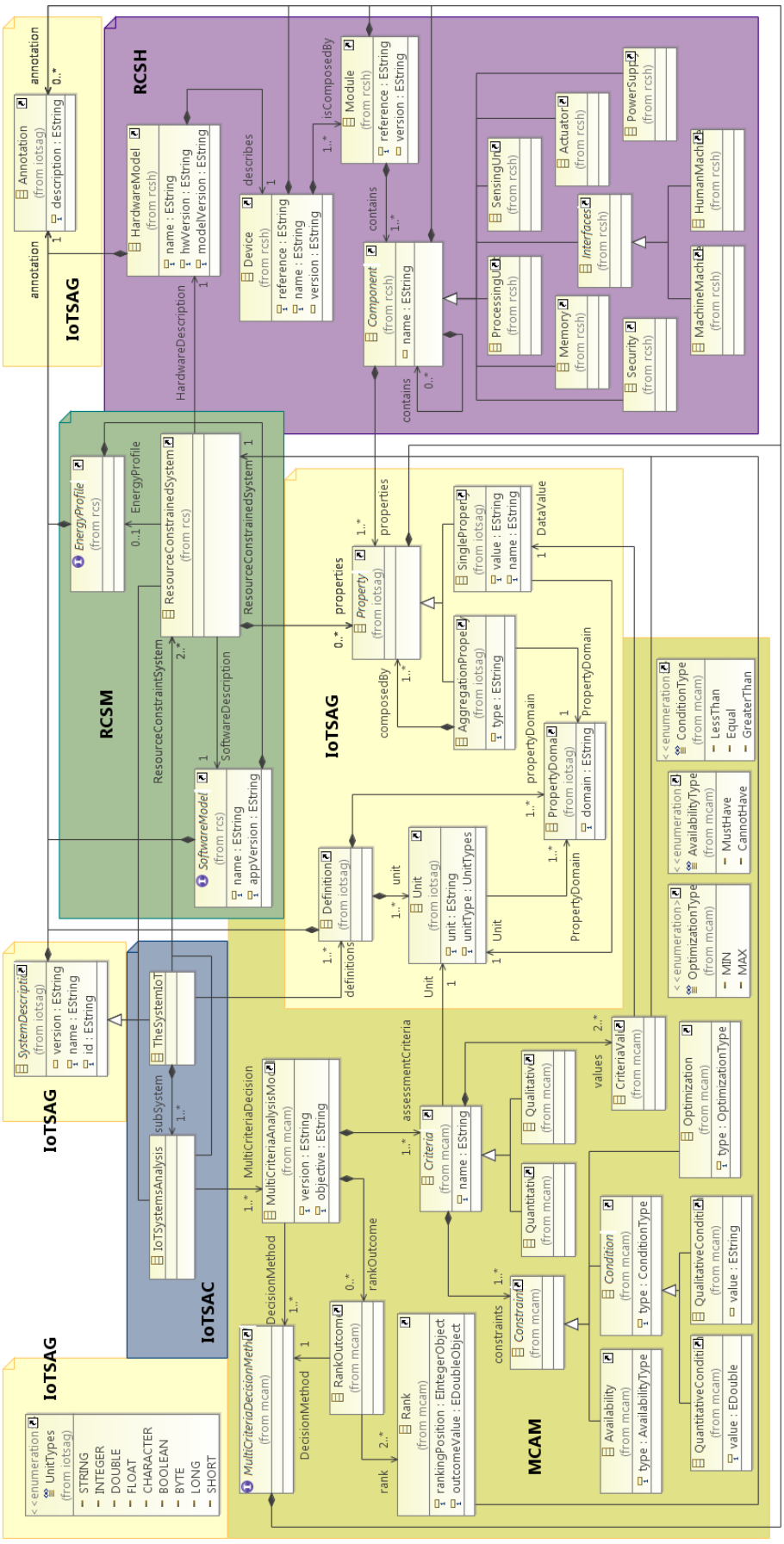


Figure 7.5: Design Support of IoT Systems: Main Specification Models.

7.4 Harmonisation Layer & Interoperability Engine

Harmonisation Layer is one of the main blocks of the proposed Multi-Criteria Framework for Design Support of IoT Systems. It is responsible of interconnect all blocks and components, maintain interoperability, and execute necessary facilitators (***Enablers***) to enable data exchange across the framework and with other systems. The ***Harmonisation Layer*** follows a Model-Driven approach based on MDA that provides generative and transformational techniques for software engineering, system engineering, and data engineering [164, 165]. A Model-Driven Harmonisation Framework was presented in Section 5.5 and depicted in Figure 5.14, the same principle is followed in ***Harmonisation Layer*** of the overall framework, introduced in this chapter.

Next it will be addressed pre-established facilitators (***Enablers***) focusing first on data, information sources regarding devices hardware, its common forms of availability, followed by method to create new IoT Systems using different and already formalised hardware components. Then, interoperability with standards is addressed, emphasizing well-known modelling languages, proceeded of the possible trade-off between IoT Systems formalisation and energy consumption. It finalises with the ***Enabler***' presentation, ***Service Bus Interface***.

Available information regarding hardware components of IoT Systems is very technical and wide, making it very difficult to analyse all aspects by human beings. Besides, markets are offering a wide set of different devices, contributes even further to aggravate this problem. Manufacturers distribute device's technical information in their websites, datasheets, or through modelling descriptions and systems. In [14, 15] stakeholders have the possibility to select and collect information data depending on the performed query, using a web page interface to retrieve data from manufacturers repository. Also, information can be provided by stakeholders (e.g.: end users, developers) but normally from DiY scenarios, small laboratory test-beds, etc. In addition, technical information of IoT Systems physical components can be scattered all over the Internet (due to a broad number of manufacturers), when it is considered built a new IoT System with multi-functionalities based on different but already available market solutions.

To address these issues, the ***Harmonisation Layer*** by executing the right ***Enablers*** provides means to identify a concrete set of data, reasoning over retrieved data, and consequently transform, fulfil the proposed formalisations of IoT Systems. To accomplish this, several types of ***Enablers*** can be used or created. Identified are: ***Ontology Markup***, based on semantic analysis (from knowledge databases — Ontologies) to match features, criteria units (units' conversion) for example; ***Merge/Transformation*** to aggregate data information from two or more different sources (following or not the same specification models), for example used when setting a new IoT System from separated definitions (e.g.: micro-controller with a Wi-Fi shield); and ***Parse Information*** based on syntactical mapping, defined rules between two different specification models (e.g. HTML, language for web-pages, to IoT System specification).

Enablers of type ***Ontology Markup*** are not here detailed, and high-level examples of ***Merge/Transformation*** and ***Parse Information*** are presented in Figure 7.6 with more practical examples addressed in next chapter.

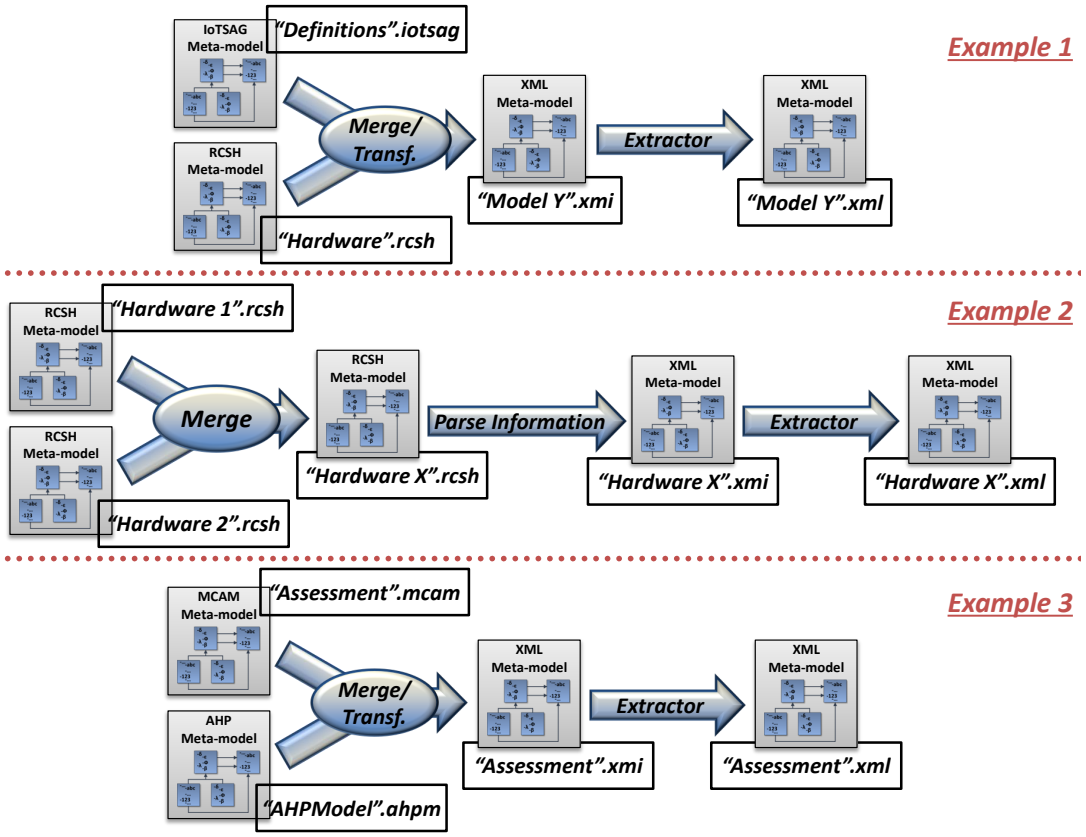


Figure 7.6: Harmonisation Layer & Interoperability Engine: Merge/Transformation Examples.

Figure 7.6 depicts three examples, possibilities involving the proposed specification models. Although, many more are possible and even more if it is considered interoperability with other specifications. The first two examples are related with the hardware characteristics of an IoT System, while the third is related to assessment of IoT Systems.

Besides the proposed specification models, Figure 7.6 presents XML and XMI languages. Utilisation is due to the fact that XML is a language widely used and that can be read by machines as well as by humans, while XMI is used as interchange format for XML files.

First example in Figure 7.6, merge, aggregates units definition (IoTSAG file) with the hardware characteristics of an IoT System, so it can be transformed to a single representation, file describing IoT System hardware components. Note that general definitions (units and properties type and domains) are kept in an independent file (a file described by IoTSAG specification model) for reusability reasons.

The ***Extractor Enabler*** is a third-party operational component, free for XML language, which transforms information described by XML Meta-Model to XML model. The inverse

process is also available and it is called ***Injector***.

The second example shown addresses the possibility to integrate, aggregate several hardware descriptions to form different solutions (IoT Systems). The hardware part of an IoT System can be formed by different modules (e.g. shields), i.e. coupling boards together. This enables stakeholders, in a clear way, to create a final hardware block. All similar to the previous example, the difference resides in one additional step to reach an XML representation. First, two hardware models are gather together by a merge operation, since both comply with the same specification model (RCSH Meta-Model), followed by a conversion between RCSH and XML syntax (rules specifying syntactic mapping — ***Parse Information***). The final step was described in *Example 1*.

The proposed IoT Systems assessment methodology enables the possibility to apply different MCDM methods, even new or user-defined methods. Section 6.4 presented formal specifications for two well-known MCDM methods (AHP and ELECTRE), suitable of being integrated, used with the proposed assessment methodology. To add a new MCDM method, it must be provided the specification model respecting the framework rules, and the respective ***Enablers*** to parse the necessary criteria data and implementation tool (also an enabler) for ***Interoperability Engine*** execution to obtain the method results. ***Enablers*** to parse information from IoT Systems specifications to MCAM specification models are also available, that follows the same principle, explain above. Figure 7.6, *Example 3*, depicts a high-level process example of going from a multi-criteria assessment (MCAM) using AHP as MCDM method to XML representation. This transformation process uses the same enablers as in *Example 1*.

The proposed Framework, depicted in Figure 7.2, highlights the interoperability with SensorML standard. However, integration with other standards is also possible by inclusion of the respective ***Enabler(s)***. Figure 7.7 shows a high-level view for the approach to achieve interoperability with standards.

Interoperability with SensorML standard is addressed in Sections 8.1.2 and 8.2.2, presenting the corresponding mapping rules between the proposed specification models and the standard, in which the ***Enabler*** is built upon. A scenario is also presented to validate this concept. This work focus on interoperability with standards built to define IoT Ecosystems requirements, behaviours, processes, etc., to serve as bridge from the design phase to IoT Ecosystem working management (real measurement, observation, data analysis, etc.).

The previous examples, presented in Figure 7.6, emphasise the use XML language although other modelling languages exist and could be considered to represent the proposed formalisations. As an example is EXPRESS (ISO 10303-11) a data modelling language to define product data aspects used to represent and exchange that same data. It also allows graphical visualisation of data using EXPRESS-G representation [205]. It is the modelling language used by the Standard for the Exchange of Product Model Data (STEP), the ISO 10303, which describes means to represent and exchange digital product information [206]. Its non-utilisation is due to its unfamiliarity to most developers,

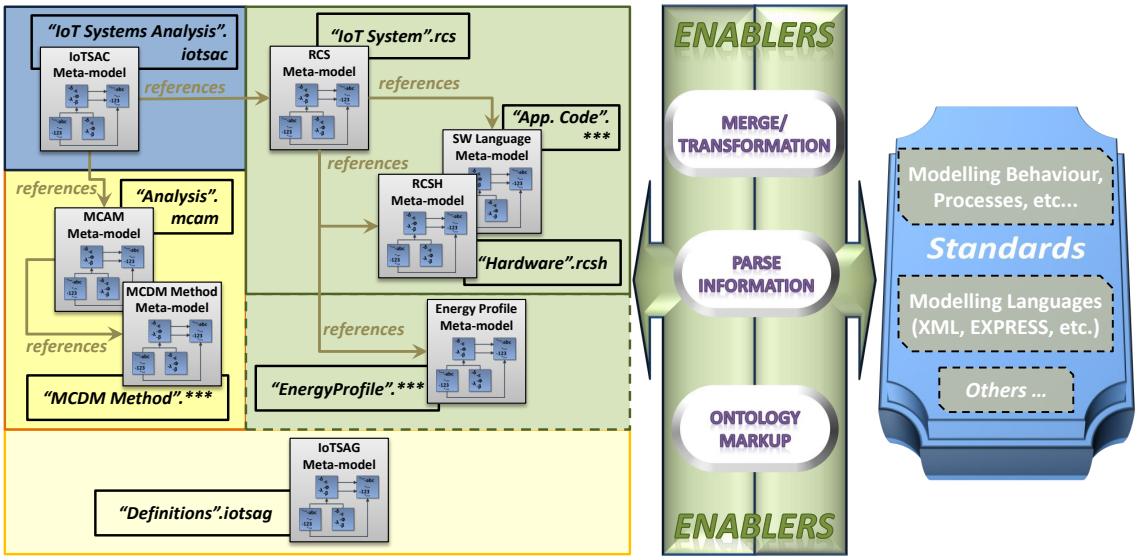


Figure 7.7: Harmonisation Layer & Interoperability Engine: Interoperability with Standards.

which poses to smaller companies a huge effort to hire or educate employers [207]. Moreover, the use UML classes to define the proposed formalisations are based on author's knowledge and common, worldwide utilisation.

Energy simulation is also an emphasized tool, system from/to which this work could be associated, contribute and/or take advantage. This topic is not envisaged as a contribution of this work and therefore will not be addressed in detail here. Nevertheless, to focus on the identified framework' openness to interoperate with other tools, systems, methods, etc., an integration process example considering a generic energy consumption evaluation tool is presented next.

Section 5.4.3 — ***IoT System: Energy Profile Formalisation*** proposes a high-level architecture from which a trade-off can be obtain between the proposed IoT Systems specifications and simulation results of energy consumption. The architecture depicted in Figure 5.10 indicated that IoT Systems formal descriptions can be used by simulations tools to build more precise energy models, contributing in this way to better, improved energy consumption information.

Considering the proposed high-level architecture presented in Figure 5.10, the envisaged behaviour within the ***Harmonisation Layer*** regarding data flow (interoperability operations achieved by applying ***Enablers***) is depicted in Figure 7.8.

On the left of the mentioned figure is presented the IoT System formal description, while on the right is depicted a generic ("black box") energy consumption evaluation tool. The ***Energy Analysis*** can provide or not its specification model (Meta-Model), in which simulations are based. However, to achieve interoperability between the two systems, ***Enablers*** must be provided to execute such task, i.e. parse information from IoT Systems representation to the simulator models.

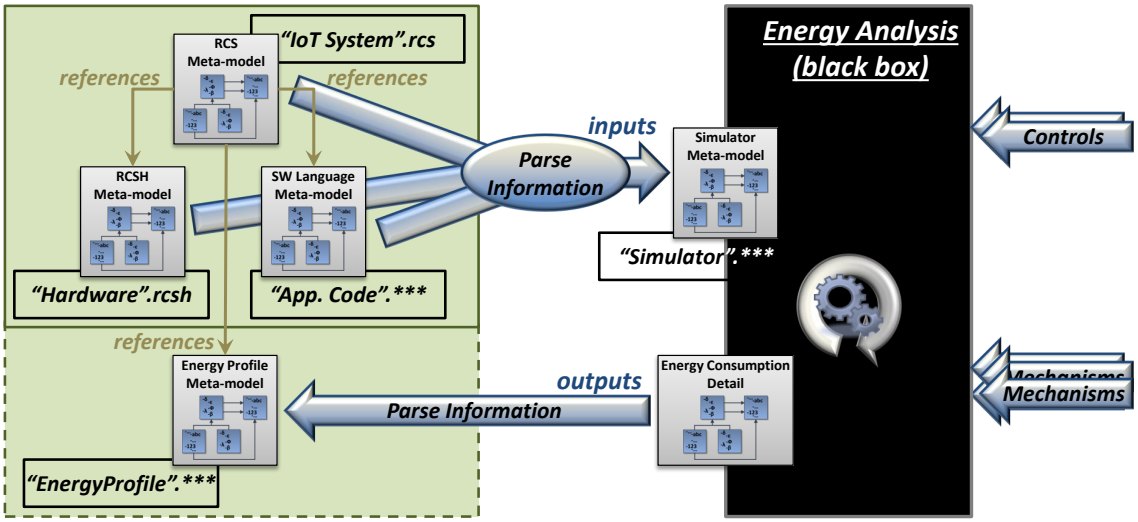


Figure 7.8: Harmonisation Layer & Interoperability Engine: Integration of Energy Simulations.

Providing the formal, detail data of an IoT System, energy simulation can be performed and the outcome (detail energy consumption) inserted into an ***Energy Profile***. This allows the completion of a formal IoT System representation. It is important to point out, that being an external system (the ***Energy Analysis*** tool) it can use a different representation language (e.g.: XML). Therefore, it might be needed an extra step to convert the data. The previous examples addressed this issue.

To finalise the more important features description provided by ***Harmonisation Layer***, it is enhance the possibility to access the assessment of IoT Systems through a ***Service Bus*** — an *IoT Systems Multi-Criteria Decision Service*. This service enables the analysis of different IoT Systems, allowing stakeholders to improve their own or verify other available solutions by examining the performance according to a set of features. Basically it provides means to access the framework and obtain indication of the more suitable IoT System for a certain task — a response as a service.

Multi requests to the *IoT Systems Multi-Criteria Decision Service*, through a Product-Service System bus, each request with its own specifications, will allow a complete build of an IoT Deployment. Requests/responses are accessible by an ***Enabler***, the ***Service Bus Interface***, responsible for maintain interoperability between framework specification models and exchanged messages content. A message is in binary format, which can contain for example XML, or just text. Messages exchange was addressed in Section 2.3.2.2 by presenting standard mechanisms called Message-Oriented Middleware (MOM).

7.5 Topic Discussion

This chapter presented the final contribution of this research work. The purpose of this contribution is to aggregate the two previous contributions, presented in Chapter 5 and

6, and in addition, not only respond to the sub-research question, Q1.3, raised in Section 1.4.1, stating: "*Which multi-criteria decision framework would provide a suitable decision support for the design of IoT Systems?*"; but also address the identified main research problem, and consequently respond to the main research question that states: *How can Internet-of-Things Systems be designed to optimise the matching with the operating environment?*.

In this sense, it was proposed a ***Multi-Criteria Framework to Assist on the Design of IoT Systems***, that provides automatic mechanisms to create and use IoT Systems formal definitions, enabling a more conscious, aware, accurate selection of a more suitable IoT System(s).

Based on a Model-Driven approach, the proposed framework presents a ***Harmonisation Layer*** with a high openness level to include different software languages and MCDM methods. Stakeholders are free to define, formalise an IoT System with their own software language characterisation, as well as apply existing or user-defined MCDM methods within the proposed multi-criteria assessment methodology.

With mechanisms to assist during the design phase, tackling the process of perform an aware choice regarding the more suitable IoT System to execute a certain task, the proposed framework presents methods to interoperate not only with modelling languages standards but also with IoT Ecosystems' real-time management tools, standards or not.

The framework' openness to work with other tools and/or systems, unlocks a large number of possibilities. For example, integration of energy simulation tools based on well-characterised IoT Systems, or high-level visualisation of hardware components (i.e.: 3D visualisation tools), using transformations (***Enablers***) to their specification models.

Chapter 8

Implementation and Hypothesis Validation

In this chapter the proposed contributions feasibility is addressed. The proposed ***Multi-Criteria Framework to Assist on the Design of IoT Systems*** is tested using a set of Proof-of-Concept (POC) implementations, and moreover an industrial implementation is also presented, in order to validate the hypothesis presented in this research work.

8.1 Proof-of-Concept Implementations

To demonstrate the conceptual approach feasibility for design support of IoT Systems, a test-cases set was created to verify contributions potential to be used. The first two test-cases are to prove IoT Systems formalisation concept presented in Chapter 5 "*Framework to Formally Describe an IoT System*" is valid. Each one of these test-cases contains scenarios to present how a stakeholder is able to describe an IoT System. The scenarios provide a high level vision on how to formally represent the physical part (hardware) and the software language used in the design of an IoT System. It is followed by a test-case to prove the validity of the IoT Systems multi-criteria assessment concept presented in Chapter 6 "*Assessment Framework for IoT Systems*". Finally, and to consider the motivations for this work, it is important to prove that the concept presented in Chapter 7 "*Framework for Design Support of IoT Systems*" is viable for the design of IoT Systems for different areas of applicability (e.g.: smart buildings, Intelligent Transportation Systems, Industry 4.0).

Figure 8.1 depicts, and in-line with a new world — a "Smart World"— that surrounds us, that many are the application scenarios (e.g.: Smart Cities, Industry 4.0) embracing IoT technologies. Being IoT the idea of a global, dynamic network infrastructure where physical and virtual "things" (devices, sensors, smart objects, etc.) communicate and share information among each other [3], and with manufacturers engaged in developing new embedded systems to address the variety of application domains and services [32],

the number of possible physical “things” for each specific task is astronomic.

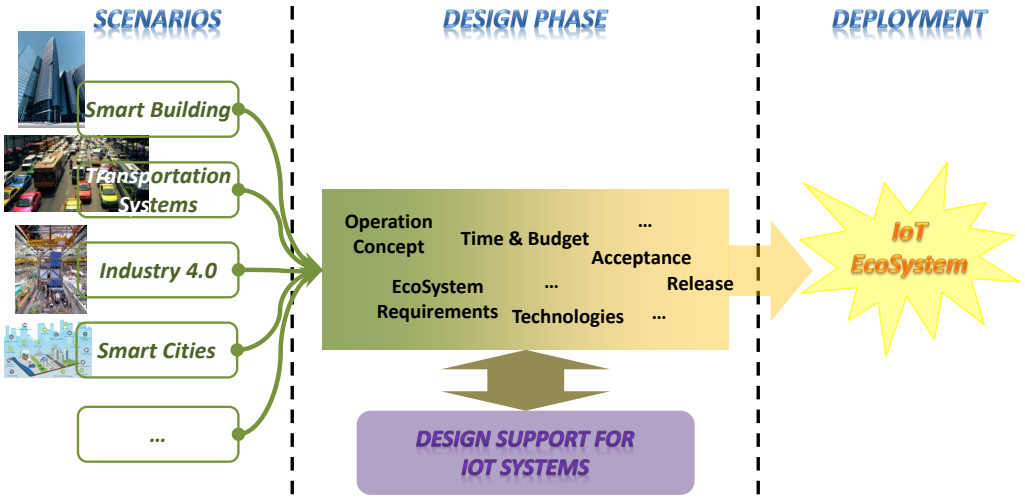


Figure 8.1: Design Support of IoT Systems: Proof-of-Concept Application Scenario(s).

Therefore, the topic design support of IoT Systems within IoT Ecosystems is applicable and interdisciplinary by nature, since it involves all Smart System scenarios. The Figure 8.1 highlights four scenarios: Smart Building, Transportation Systems, Industry 4.0 and Smart Cities; but others exist and could be mentioned as for example Smart Home, Smart Farming, Smart Health, etc. However, the focus of this thesis concerns interoperable IoT Systems assessment, thus the POC illustrates how to formally represent and then perform a conscious assessment of IoT Systems and how interoperability with other systems, tools is facilitated. Two application scenarios integrate the Proof-of-Concept (POC) and are described in the following sections:

- The first scenario addresses how IoT Systems description and assessment is achieved. Aims the necessary formalisation of an IoT System and the assessment methodology;
- The second scenario focus on the interoperability with standards, namely an IoT Ecosystems description standard;

8.1.1 Application Scenario 1: Smart Building Design

As mentioned, and depicted in Figure 8.1, it was possible to consider any scenario under the IoT scope as POC due to the sensors network nature of IoT deployments. In this way, lets consider a Smart Building scenario as presented in Figure 8.2.

A Smart Building ecosystem provides means for many kinds of applications, such as for security, fire and gas safety, lighting control, 24/7 monitoring, HVAC control, energy saving, etc., based on a continuous environment surveillance performed by IoT Systems. The scenario depicted in Figure 8.2 illustrates that hundreds if not thousands devices are place in a skyscraper building, some to perform one type of function others another

(e.g.: sensing temperature, turn-off lights, control gas leakage), all working together for a common good — people welfare. The tasks diversity within the deployment and the availability of numerous solutions, either in terms of hardware or software languages poses as an issue. Decide on which is the proper solution for each task and in what ground will solutions be selected are two topics that this dissertation addresses.

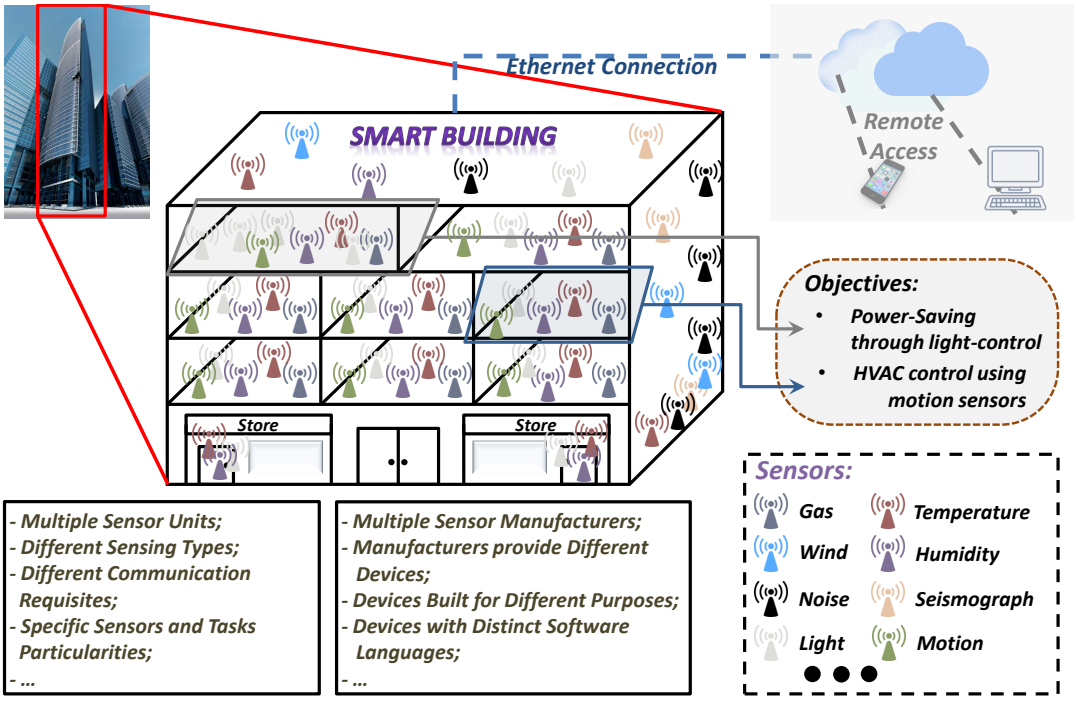


Figure 8.2: Proof-of-Concept Scenario 1: Smart Building Design.

The scenario, *Smart Building Design*, presented in Figure 8.2 depicts several application possibilities (e.g.: security, fire and gas safety). However it is impossible to consider and address all the possibilities, therefore two possible applications are highlighted: Power-Saving through light-control; and HVAC control using motion sensors. The objectives focus on getting an indication of which is/are the more suitable IoT Systems to perform:

1. **Objective 1:** Power-Saving through light-control. Light sensors, place by room-zones, sense the environment (light values) and act on the respective light modules (luminaire) controlling the light-intensity;

2. **Objective 2:** HVAC control using motion sensors. HVAC is turn-off/on accordingly to movement in room and days' time verification.

In terms of hardware platforms, a set of possible solutions are considered for both cases. Software languages are applied, chosen accordingly to hardware platforms with which they will be integrated. Multi-criteria assessment of IoT Systems methodology is apply for both objectives, but criteria, constraints, priorities and MCDM methods are considered separately, accordingly with respective objective.

To demonstrate how the IoT Systems description and assessment is achieved (presented respectively in Section 5.2 "Framework for IoT System Formal Description", and Section 6.2 "Framework for IoT Systems Assessment"), the POC for this scenario was split in three test-cases and are presented in next subsections:

1. How to formally represent IoT System physical components (hardware);

2. How to add formal representation of software languages to the framework and associate it to the formal description of an IoT System;

3. How to perform IoT Systems assessment and obtain a more suitable solution for a specific task.

8.1.1.1 IoT System: Hardware Components

Formal descriptions of IoT Systems are crucial not only for this work but also to other existing design systems or new ones. The ability to define operations based on specific data structures enables processes to be automatic and easily upgradable. The definition of IoT Systems characteristics, features is the groundwork for the main contribution of this work. As mentioned, manufactures are providing several solutions to act as IoT Systems and many of them with expansion capabilities (i.e. aggregate, join different hardware parts to form a single hardware). However, the information regarding these solutions is scattered all over the internet, where specifications are normally provided through web pages, datasheets (normally in PDF), user-defined text, etc.

Figure 8.3 presents a general perspective of how to go from scattered information of a large diversity set of hardware solutions to a specific, formal description of these same solutions.

At the top is represented some hardware alternatives, of a market with a much larger offer. Manufactures are focus mainly in providing two types of boards/devices. First devices with a set of built-in sensors and wireless communication, but with no capacity to be expanded, i.e. impossible add new hardware features to these devices (referred as specific boards). Second, development boards, larger in size, normally with no sensor or communication features (from scratch), but with a more user-friendly development environment and with the capability to be expand with other hardware components (e.g. communication boards, dc motors control boards, sensor units, etc.).

To get to the bottom, to the IoT Systems formal descriptions, is necessary to gather the hardware information and convert it to the right specification models. Stakeholders have the opportunity to identify new components or combine some if possible. It is also possible to reuse formal descriptions already available or combine these with existing or new boards.

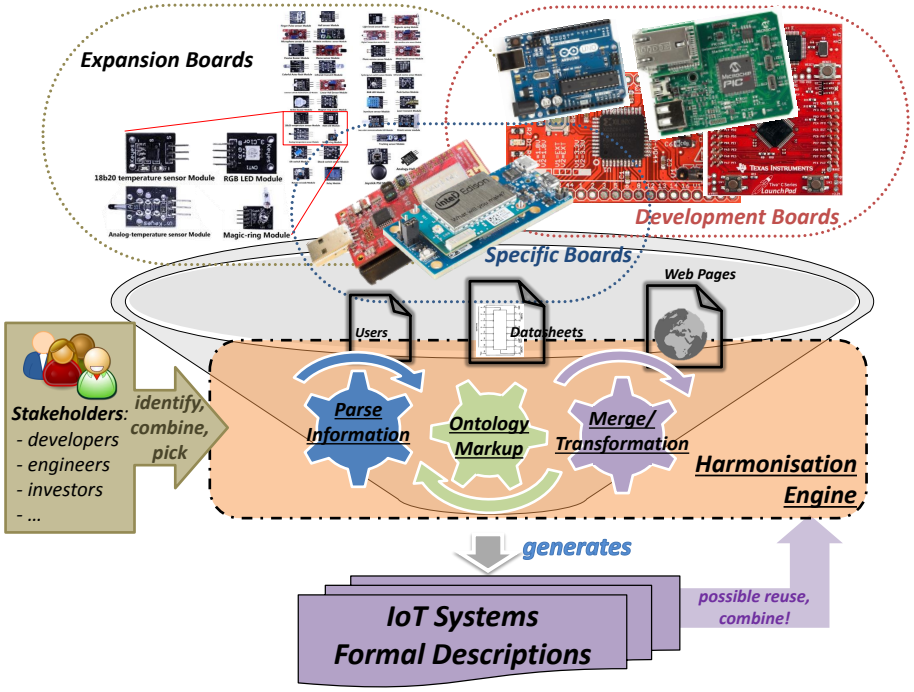


Figure 8.3: From Scattered Hardware to Formal IoT Systems Specification (Hardware).

8.1.1.2 IoT System: Software Languages

The number of software languages available is high and in most cases they do not share a common ground for the rules they obey. As POC for the integration of a software language into the proposed framework and consequently allowing to build a properly formalisation of an IoT System, next is presented two software languages adaptations. These programming languages are the C language and the nesC.

To properly integrate a software language two rules must be obeyed by the specification model. Additional information in the form of properties must be specified using IoTSAG specification and the main model class inherit from the interface class *SoftwareModel* from RCSM specification.

C Language Specification

The C Programming Language [117] is a widely used software language. As mentioned in Section 2.3.4 this language is used by two of three most known and used OS in the IoT. Figure 8.4 depicts a specification model for C Language version 1.0 retrieved from [208], adapted to be part of an IoT System specification. Meaning that a software application, can be described using this language and associated to the respective IoT System.

The Ecore representation of the adapted C Language specification model is presented in Appendix H.

The specification model of the C Language suffered one necessary change to fulfil the rules imposed by the proposed framework. This is the addition of class *ApplicationModel*, inheriting from class *SoftwareModel*, to the Meta-Model.

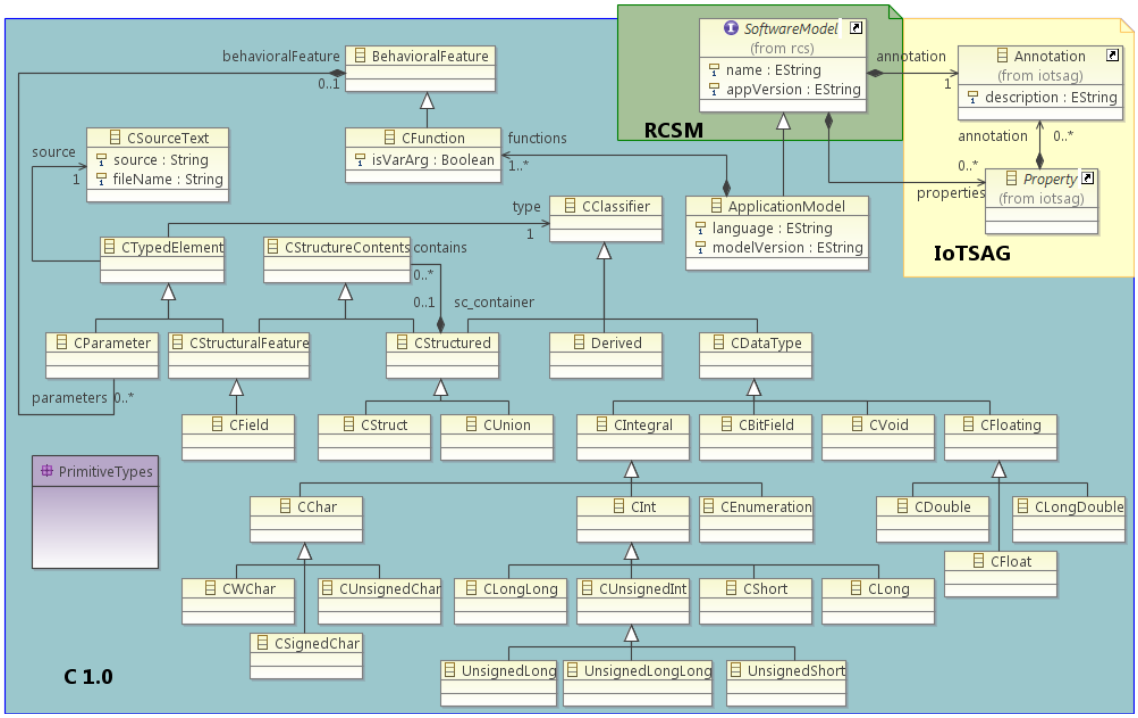


Figure 8.4: C Language Specification Model (version C1.0 adapted from [208].)

Class *ApplicationModel* was added as main model class, including two attributes and one class reference. The attributes, *modelVersion* and *language*, indicate the model version (for version control) currently at "1.1-2019" and the software language name set to "C Language", respectively. The class reference, *functions*, pointing to class *CFunction* from the original specification, is used to reference at least one function of C Language programming file. As known, a C file contains at least one function, the main function. The detail composition of C Language specification is not presented since it is not the focus of this work to explain the constructs of this model, but rather to explain how to add a software language to the proposed framework.

nesC Language Specification

The second software language used as POC for the openness of the framework to use different programming languages is the Network Embedded Systems C (nesC) [109]. The software language of TinyOS the OS more used by the scientific community accordingly to the study presented in [128]. The nesC specification model shown in Figure 8.5 contains the necessary changes so it can be included in the framework. Changes made to the nesC Meta-Model proposed by João Ralo in [209], a specification model to describe and formalize the nesC programming language, a work supervised by the author. This work was developed based on the reference manual of nesC language version 1.3, written by David Gay [210], and on the formalisation proposed by Rontidis Pavlos [211]. Although, this last formalisation was incomplete since it only describe part of the nesC Language.

The Ecore representation of the nesC Language specification model is presented in

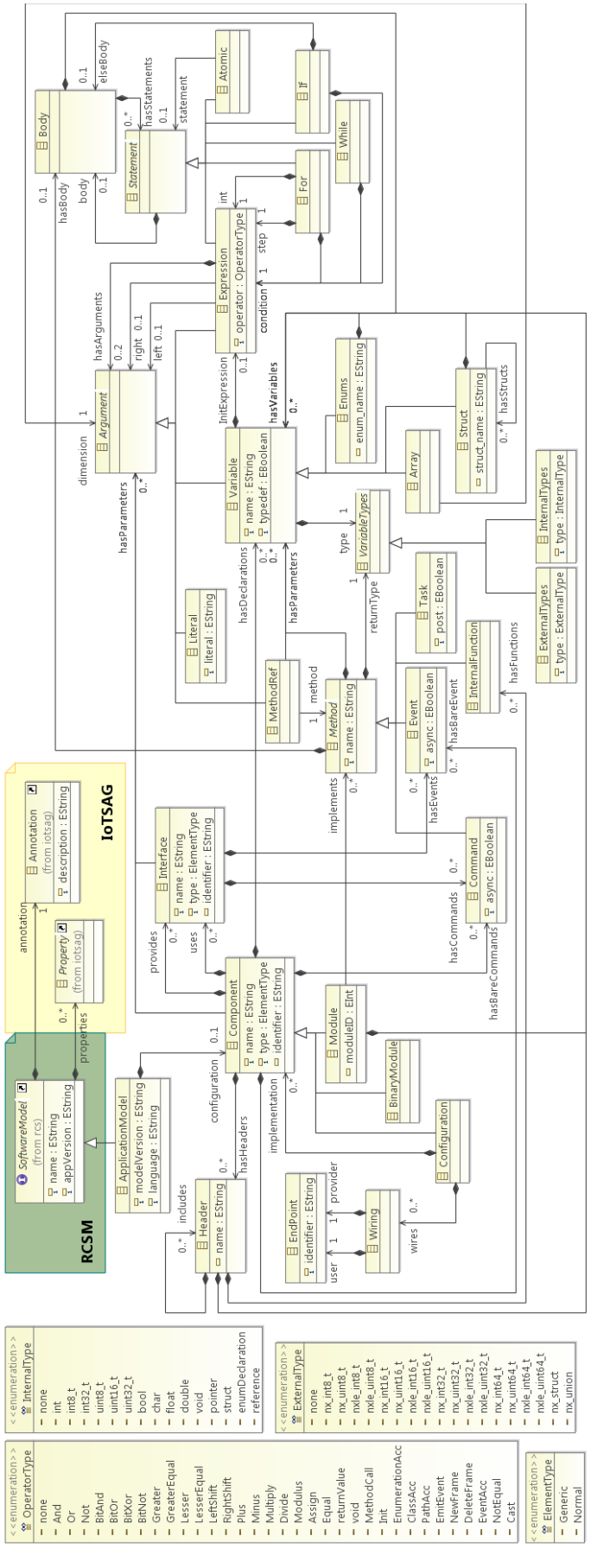


Figure 8.5: nesC Specification Model (adapted from [209]).

Appendix I.

The necessary change to nesC specification model was to add *SoftwareModel* interface class as parent of the core class *ApplicationModel*. This class already provides means to describe the language name and model version, with pre-defined values of "nesC" and "2.0-2019", respectively.

8.1.1.3 Multi-Criteria Assessment of IoT Systems

The third, and final test-case considered for the *Smart Building Design* scenario is the POC of multi-criteria assessment of IoT Systems to obtain the more suitable solution for each specified task/objective. Figure 8.6 shows a high-level view of that decision process.

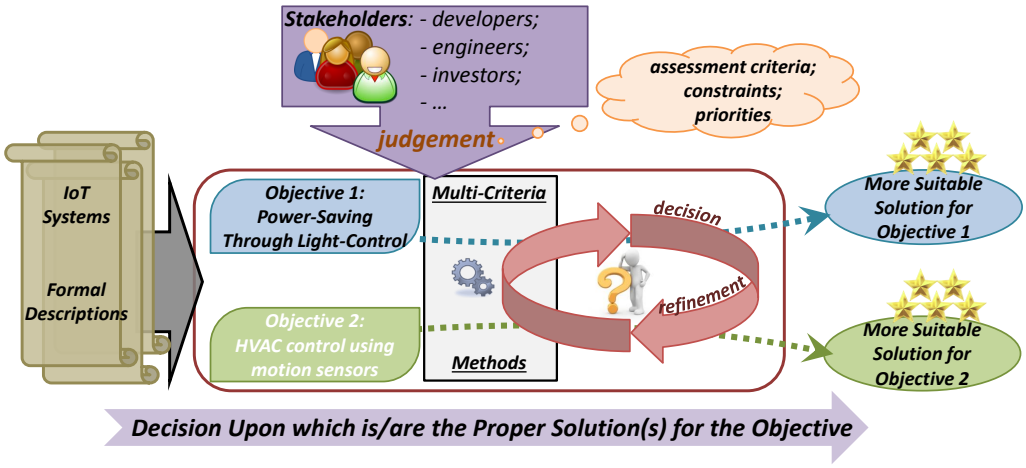


Figure 8.6: IoT Systems Assessment: Two Objectives of Smart Building Design Scenario.

The IoT Systems are described in terms of their hardware features; software language; and respective application code, and are all considered as possible solutions for both objectives. However, stakeholder's judgement is not the same for the two cases. In this sense, for the first objective, *Power-Saving Through Light-Control*, it is considered that the solution must contain a light-sensing unit, a PWM output signal to control the light module and with a low price. The second objective, *HVAC Control Using Motion Sensors*, dictates that to be used in this case an IoT System must be able to communicate using IEEE 802.15.4 protocol for time synchronisation, at least one digital output for on/off relay control (HVAC on/off), a motion-sensing unit and it has to present a good quality/price relation (low cost). It is considered that the relay module it not included for analysis.

8.1.2 Application Scenario 2: SensorML Standard

The Sensor Model Language (SensorML) [21], is an OGC standard used to describe sensors functional models with the capability to represent components as processes, physical (e.g. detectors, actuators) as well as non-physical (e.g. mathematical operations or functions). Sensors and sensor systems are defined using geometric, dynamic, and observational characteristics.

SensorML standard uses models and a XML encoding to describe processes. These processes, which are discoverable and executable, define their inputs, outputs, parameters, and method, as well as provide relevant meta-data. It can work as an electronic specification sheet perform sensors, sensors systems and process discovery. Provides support for tasking observation and alert services, with also an on-demand processing of Observation outcomes among other features. Figure 8.7 presents the SensorML main packages dependencies, retrieve from [21].

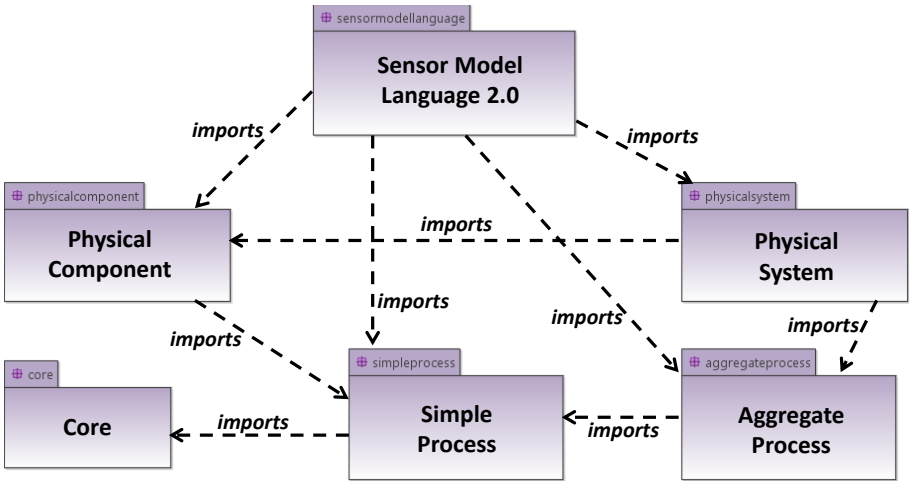


Figure 8.7: SensorML Main Internal Packages Dependencies (retrieved from [21]).

The proposed framework and specification models presented do not intend to replace or be concurrent, better than SensorML standard, but rather assist, work together, automatize the process that goes from the design phase to the real measurement and observation processes within an IoT Ecosystem. The scenario presented in Figure 8.8, depicts an example, "Internet of Things — Simple Sensor", retrieve from [212], the SensorML examples database. In this scenario is described a sensor with a simple data stream containing temperature values. This sensor is an online processing component (simple thermometer) with a link so real-time data can be obtained.

On the left of Figure 8.8 is depicted a generic sensor acting as *simple thermometer*. At the middle, SensorML model classes are identified accordingly to their use in XML representation of the considered example, on the left. A complete version of the XML file describing the entire example can be found in Appendix J.

Regarding the POC of this scenario, it is defined a test-case to demonstrate the interoperable nature of the proposed framework. The test-case implementation consists on providing the mapping between SensorML and proposed specification models, enabling in this way the process to go from a decision at design phase, of which is the more proper IoT System, to the objective fulfilment — online temperature observation.

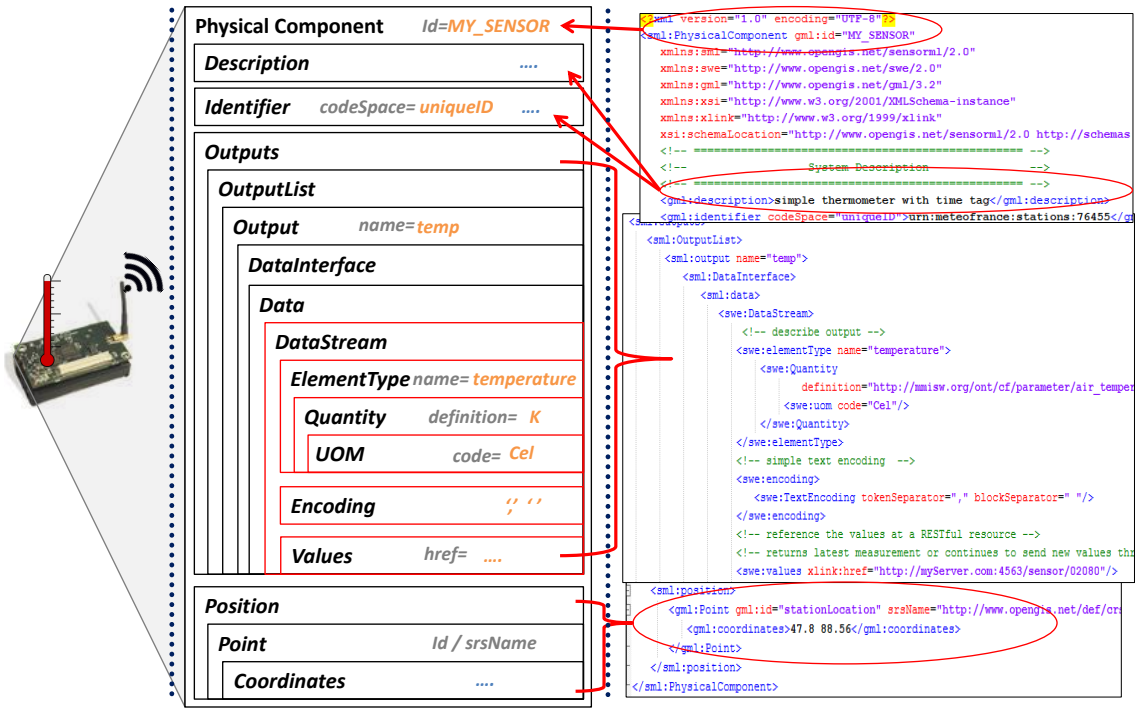


Figure 8.8: Proof-of-Concept Scenario 2: Temperature Sensor with Online Data Observation (SensorML Example).

8.2 Technical Implementations

Section 8.1 of this work presented as Proof-of-Concept (POC) a set of test-cases scenarios that addressed the implementation of Chapters 5 to 7 frameworks and architectures specification. However, a POC is not supposed to be flawless, robust or a complete solution, instead it is intended to demonstrate the feasibility of a solution and/or idea.

Therefore, and as proof of feasibility, next technical implementations are presented for each of test-cases proposed, and consequently verify the contributions potential.

8.2.1 Implementation of Scenario 1: Smart Building Design

The Smart Building Design scenario presented in Section 8.1.1 and depicted in Figure 8.2 addressed two objectives: first, energy saving through by controlling light modules; and second a HVAC on/off switching using motion sensors. This scenario was in three test-cases matching the contributions: IoT Systems hardware formal representation; associate application code to an IoT System; and perform an assessment and decision regarding the more suitable IoT System(s) for the design objectives. The next sections aim to demonstrate the contributions potential for each one of these test-cases.

8.2.1.1 Hardware Description of IoT Systems

The proposed IoT System formal specification (Resource-Constrained System Meta-Model (RCSM) represented in Figure 5.5) states that to correctly and completely describe these

systems three parts must be considered, even if one is not mandatory. The two core parts are **hardware** and **application code** formal descriptions, and the optional relates to **energy consumption**. IoT Systems hardware formal representation from scattered, unformatted data sources is presented next by describing implementation steps of the MDA-based Harmonization Framework.

As it would be impossible to include all hardware components description, a selection was made upon two platforms: an AdvanticSys[34] device (CM4000) and the Arduino[35] Uno (already used as an example in Section 5.4.1).

Regarding the proposed IoT Systems specification is important to highlight, clarify some points from a technical, practical point of view. From one hardware specification (and if the physical device allows it) it is possible to aggregate other hardware specifications and consequently generate different, numerous IoT Systems specifications (possible solutions).

One possible source of data information regarding hardware components and its details are web pages, which are described using HTML. Manufacturers commonly announce their offers with a set of features to enable stakeholders to have an overview of what is being sell. Although it is important emphasize that web-pages do not have detail and complete information, but links to datasheets and technical reports are usually available. Figure 8.9 gives an high level overview of what happens when the proposed framework is used to obtain IoT Systems formal description from web-pages data source.

On the left of Figure 8.9 is depicted the two hardware platform web-pages, with a user common perspective followed by source code (HTML code). From the common user-perspective, hardware and software characteristics are described and by using the model-driven framework is possible to generate the IoT Systems formal descriptions, as represented in the right side of the figure. On top, from left to right, is presented the process for CM4000 AdvanticSys mote, and on the bottom for Arduino UNO.

Considering web-pages as data information source regarding hardware components, meaning that information is arranged using HTML specification, is possible to describe the transformation process within the model-driven framework, which deals with the process of going from a HTML representation to a RCSH and IoTSAG representation, i.e. to a formal specification of an IoT System. Figure 8.10 shows this process.

With the hardware information following HTML specification (a XML serialised) it cannot be directly put into the ATL toolkit, since this toolkit needs a XMI representation as input. Consequently, the first step, ***XML injection***, is to convert data to a XML MOF meta-model specification. Accomplished this, next follows a step responsible to map the obtained XML format data to the reference HTML Meta-Model which is the input for the actual language mapping (from HTML to RCSH and IoTSAG representations). This transformation is performed accordingly to a pre-defined mapping between the structural formats of the two languages (XML and HTML). It is a mapping between both

[34] https://www.advanticsys.com/
[35] https://www.arduino.cc/

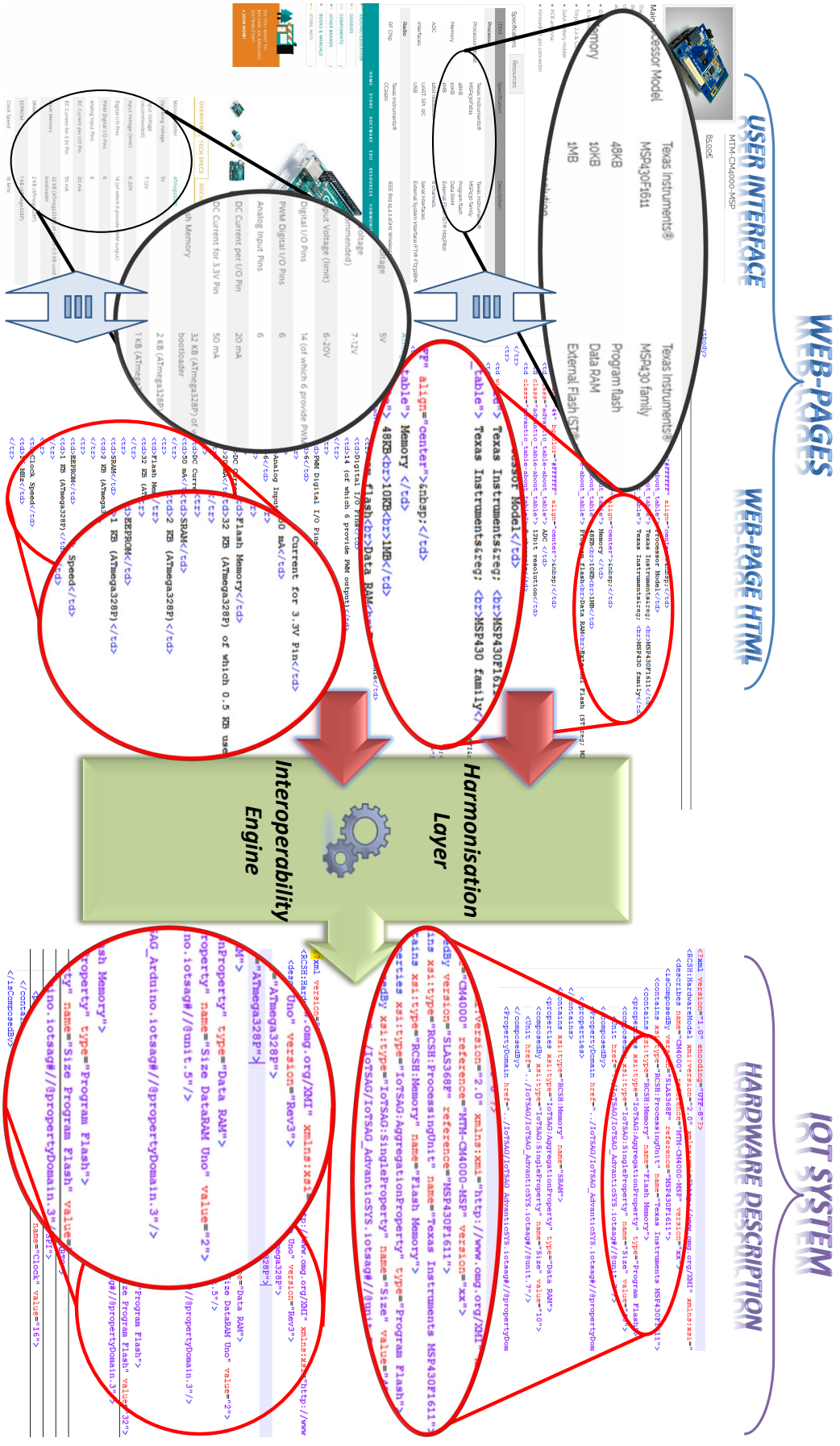


Figure 8.9: IoT Systems Formal Descriptions Based on Web-Pages Information.

specification models, i.e. Meta-Models, therefore executed at the second abstraction level accordingly to MDA approach.

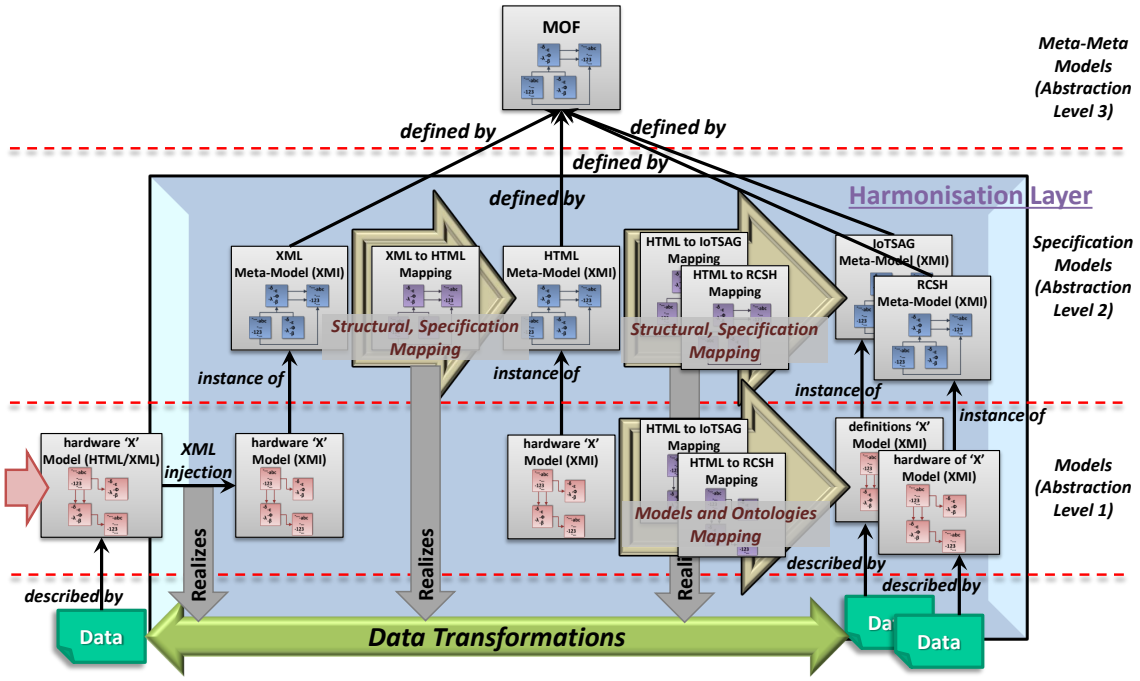


Figure 8.10: Harmonisation Layer: IoT System Physical Components Instantiation.

The transformation process, mapping from HTML to the RCSH and IoTSAG specification models is accomplished at two levels accordingly to MDA abstraction levels, more precisely, at the first and second abstraction levels. At specification models level (abstraction level 2) mapping is defined by matching input classes (HTML classes) with output classes (RCSH and IoTSAG classes), delineating in this way part of the transformation process.

The second part is realised by taking into consideration the data (mapping definition at abstraction level 1). Since XML is a generic data representation, data in each attribute, element must be analysed for a correct matching, and consequently transformations can be performed to destination models. For instance, words like processor, micro-controller, CPU are linked to the ***ProcessingUnit*** class of the RCSH specification model.

8.2.1.2 Application Code for IoT Systems

An application code, firmware of an IoT System is described by a ***Software Language*** which is the second core part to completely describe IoT Systems. Sections 8.1.1.2 and 8.1.1.2 presented two software languages specification models, the C and nesC languages, with the necessary changes to respect the imposed framework rules.

IoT Systems firmware, applications code are inserted into devices by the use of specific compilers or Integrated Development Environment (IDE) tools. These tools, normally, use text files as input developed, created by developers, programmers. The use of a

Model-Driven based framework facilitates the code generation, and consequently the deployment of IoT Systems.

Similarly to the reuse of hardware specifications to create new IoT Systems, the association of an application code with different hardware platforms (of a similar nature, e.g.: same processor family) also enables the creation of multiple, different IoT Systems.

Figure 8.11 depicts the process of going from a software language specification model to application code text files. The two software languages follow the same principle, therefore C and nesC languages are represented in a general way, "***App. Language***", in the figure. In terms of number of steps, generate text files from specification models is simpler than to perform a transformation between models. From a perspective of code generation, once the software language model is filled, with the rules established by the specification model respected, is possible by applying the necessary conversion rules obtain the application code file(s).

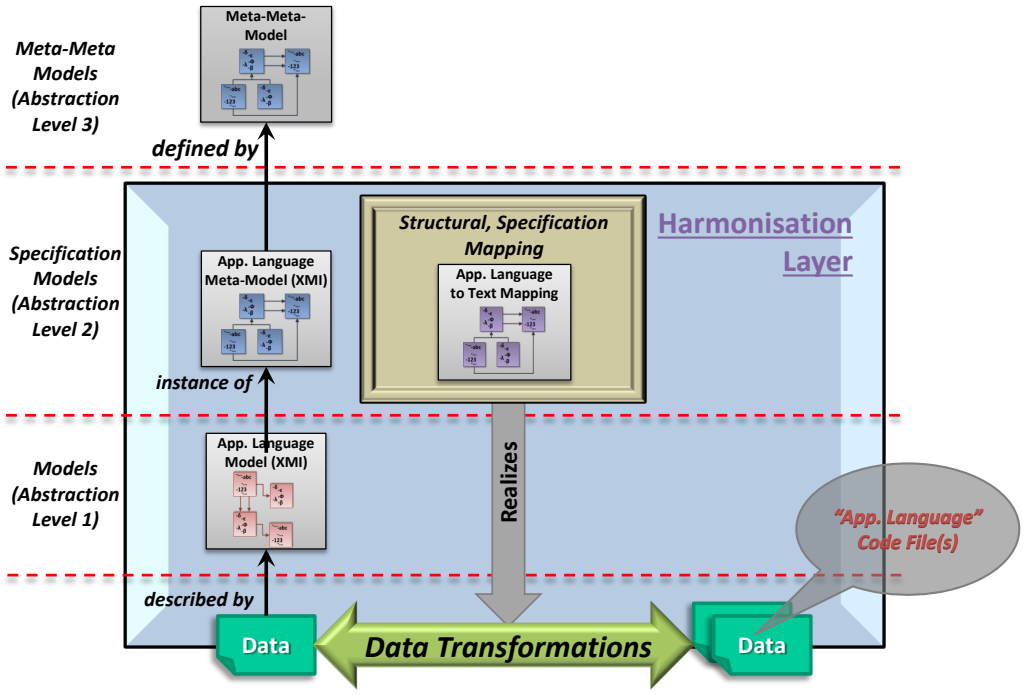


Figure 8.11: Harmonisation Layer: From Software Languages Specification to Applications Code (Text Files).

The transformation process is bidirectional, which means that it is possible to create applications code from the specification model as well as to get meta-model instantiation from code files. It will be necessary to create the rules, specify the transformations to implement both transformations.

Let's consider the nesC language as the software language used to program the IoT Systems with the necessary firmware. To execute the transformation, from the nesC specification model to code files, is necessary to implement the mapping rules. Transformations are normally static processes, specified during the system design, that once

defined can be repeated indefinitely. Mapping rules defined to be applied based on structural aspects, based on specification models produces the same structural output independently of content. Figure 8.12 depicts the process of going from an instantiation of nesC meta-model, i.e. the definition of an application, to an IoT System programmed with the application. The instantiation of nesC meta-model is identified in the as ***Step 1.0***.

In nesC case, there will be at least two necessary programming files, with ".nc" extension, for the configuration and module. It can also contain library files, ".h" files, also generated by the specified transformation rules. The transformation language used was Atlas Transformation Language (ATL). The mapping rule to create the mentioned files is presented in Figure 8.12, identified as ***Step 1.1*** (all underlying functions are not described in the figure).

Executing the transformation, ***Step 2.0***, the nesC files are created: configuration, module and 1 library file. Programming is described in the figure as ***Step 3.0***, and it was used the Ubuntu 12.04 with TinyOS version 2.1.2 installed, to compile and program the devices. A correct compilation serves as validation of the process of going from the nesC specification model to actual code files.

8.2.1.3 IoT Systems Assessment & Proper Solution(s)

Previously, IoT Systems formal description was addressed by going through the implementation aspects, explaining how data could be retrieved and aggregated from external sources, or the use of different languages could be applied within the proposed framework. With IoT Systems well defined, where information follows specific rules, tools can retrieve and/or analyse each one of the IoT System features, characteristics. These brings the third test-case defined for the Smart Building Design scenario — perform an assessment and decision regarding the more suitable IoT System(s) for the design objectives.

Let's focus on the formal representation, i.e. the specification model (MCAM) proposed in Section 6.3, that describes an IoT Systems assessment rather than on the assessment methodology proposed in Section 6.5 that uses, is based on the proposed MCAM. The implementation aspects of the assessment methodology will be addressed in Section 8.3.2 with the analysis of an Industrial scenario.

The third test-case for the *Smart Building Design* scenario sets two objectives. *Objective 1: Power-Saving Through Light-Control* establishing as criteria a light-sensing unit, a PWM output signal to control the light module and a low price. On the other hand, *Objective 2: HVAC Control Using Motion Sensors* specifies IEEE 802.15.4 compliant protocol as communication, one digital output for HVAC on/off, a motion-sensing unit and a low cost as criteria.

From the objectives definition is possible to define constraints for each criterion. The list of criteria applied to each criterion is depicted in Equation 8.1 for *Objective 1*, and in Equation 8.2 for *Objective 2*, where each line represents jth criterion. Respectively:

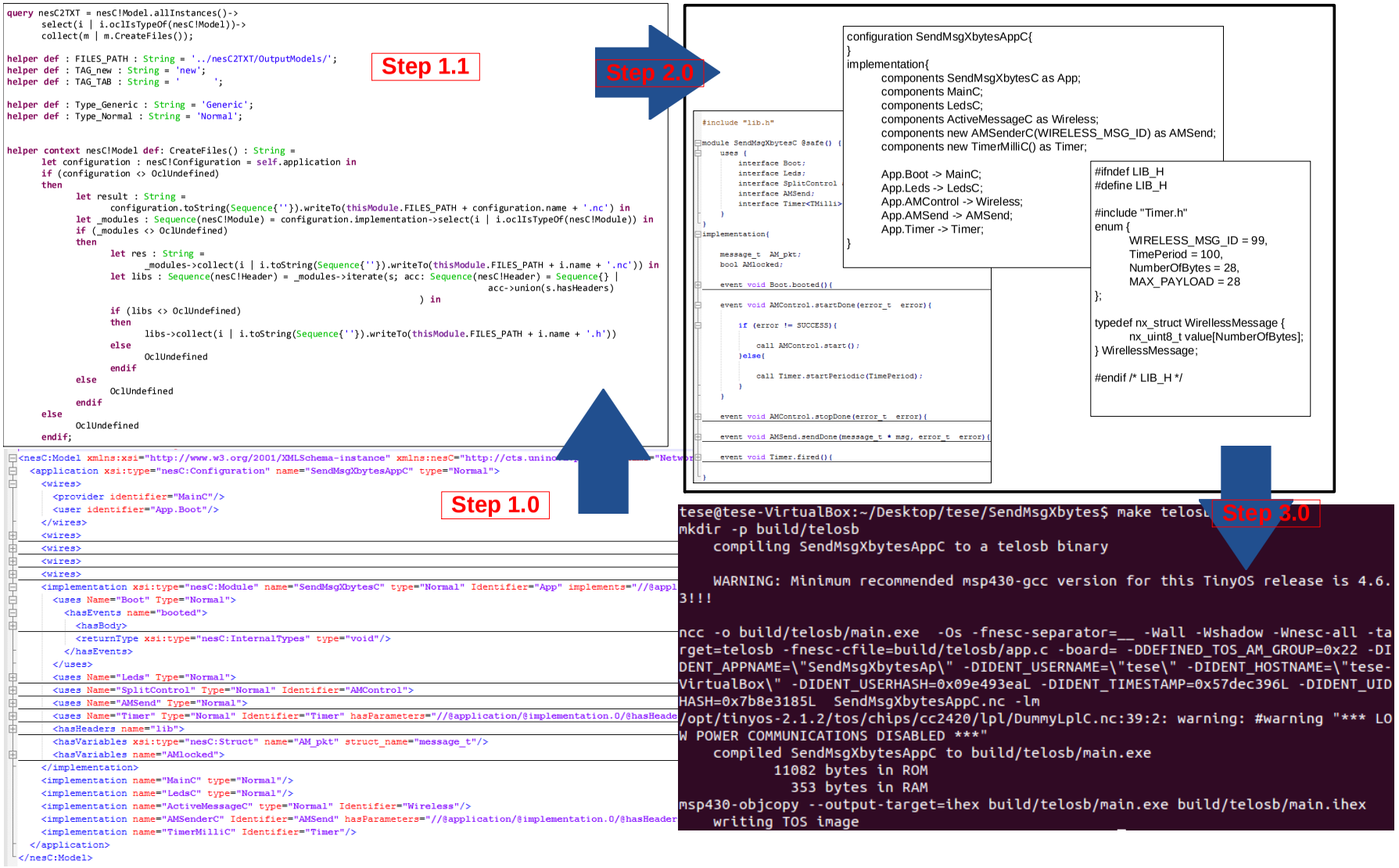


Figure 8.12: From nesC Models to Application Code (Text Files).

Light-Sensing Unit; PWM Output Signal; and Price for *Objective 1*. For *Objective 2*: IEEE 802.15.4 Protocol; Digital Output Signal; Motion-Sensing Unit; and Cost. Considering the first objective, the first line of Equation 8.1 indicates that constraint, ***MustHave***, is applied to criterion Light-Sensing Unit.

$$Constraints_{Obj1} = \{ \begin{array}{c} MustHave; \\ GreaterThan; \\ MIN \} \end{array} \tag{8.1}$$

$$Constraints_{Obj2} = \{ \begin{array}{c} Equal; \\ GreaterThan; \\ MustHave; \\ MIN \} \end{array} \tag{8.2}$$

The formal description of the first objective is depicted in Figure 8.13. Following the proposed MCAM specification it is highlighted the constraints definition for each criterion, the criteria units' definition and the indication of the MCDM method used. On the left of Figure 8.13 is presented the MCAM Meta-Model instantiation (objective 1 MCAM model), and on the right the IoTSAG Meta-Model instantiation.

Figure 8.13: MCAM Specification Model: Instantiation of Objective 1 — Power-Saving Through Light-Control.

Reading from the top of MCAM model, criterion *"Light-Sensing Unit"* presents a constraint of type *"MCAM:Availability"* with a considered unit of type *"Number"*. The no definition of the availability type indicates that the default value is applied, i.e. ***MustHave***. The assessment methodology will process this criterion looking for values different from

zero (as defined in procedure presented in Figure 6.6) to certify the possible solution as valid. Next is criterion *"PWM Output Signal"*, with *"MCAM:QuantitativeCondition"* of type *"GreaterThan"* defined as constraint (default value for this constraint is *"LessThan"*), with a considered unit of type *"PWM"*. Default value for attribute *"value"* of class *"MCAM:QuantitativeCondition"* is 0.0. The third and final criterion considered for *Objective 1* is *"Price"*, for which was established an *"Optimization"* constraint with type *"MIN"* (default type for this constraint). The MCDM method used is specified in the final of MCAM model, by the *"DecisionMethod"* tag. For this objective was considered the AHP method.

MCAM, IoTSAG and AHP models are instantiated in different files, as well as the possible solutions, IoT Systems models (RCS, RCSH, and software language). In case of being used different IoTSAG files, aggregation is performed by ***Harmonisation Layer*** of the proposed Framework (presented in Figure 7.2).

The formal description of the second objective is depicted in Figure 8.14. Following the proposed MCAM specification it is highlighted the constraints definition for each criterion, the criteria units' definition and the indication of the MCDM method used. On the left of Figure 8.14 is presented the MCAM Meta-Model instantiation (objective 2 MCAM model), and on the right the IoTSAG Meta-Model instantiation.

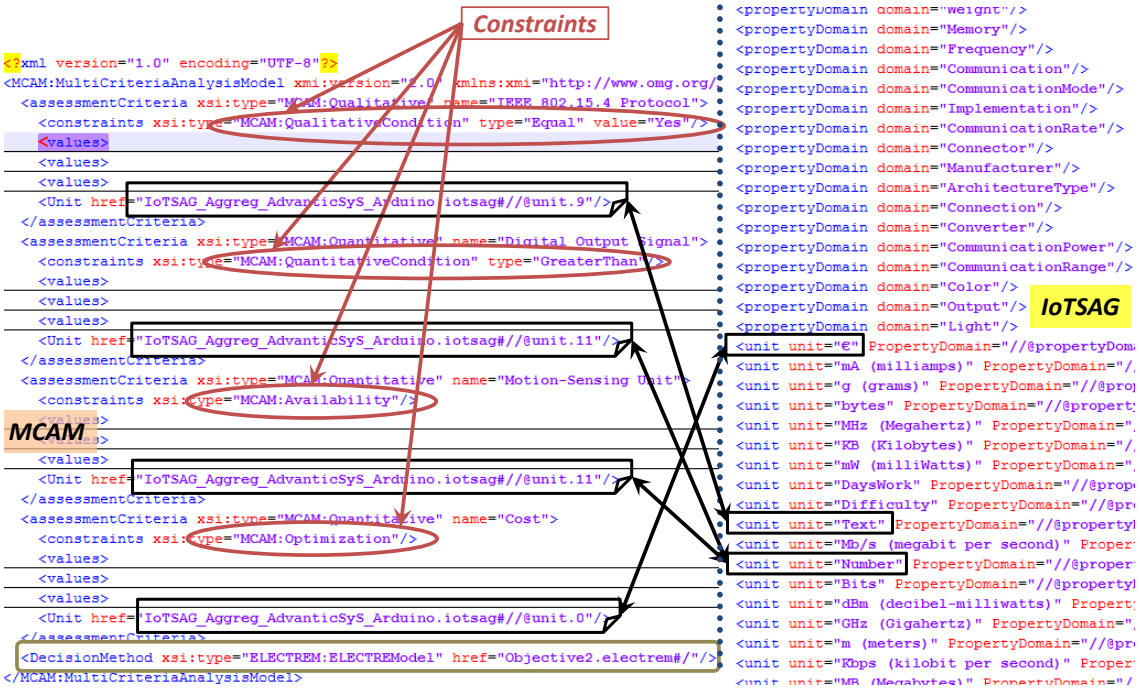


Figure 8.14: MCAM Specification Model: Instantiation of Objective 2 — HVAC Control Using Motion Sensors.

Reading from the top of MCAM model, criterion *"IEEE 802.15.4 Protocol"* presents a qualitative constraint of type *"MCAM:QualitativeCondition"* stating that criteria values (text values) have to be equal to *"Yes"*. With this assessment constraint the proposed

assessment methodology can exclude any solution that does not have a wireless communication using the IEEE 802.15.4 standard. Next is criterion *"Digital Output Signal"* with a constraint *"MCAM:QuantitativeCondition"* of type *"GreaterThan"* (default value is 0.0), establishing a condition that allows the methodology to exclude any solution that does not present at least 1 digital output pin. The two last criteria *"Motion-Sensing Unit"* *"Cost"* are in all equal to criterion *"Light-Sensing Unit"* and *"Price"* of the previous objective. The MCDM method used is specified in the final of MCAM model, by the *"DecisionMethod"* tag. For this objective was considered the ELECTRE method.

In order to address the specification models proposed in Section 6.4, which describes two MCDM methods, AHP and ELECTRE, using a model-driven approach, the scenarios presented here used these MCDM methods. Figure 8.15 depicts the methods instantiation, where on top is presented AHP used in the first objective and on the bottom ELECTRE method used for the second objective.

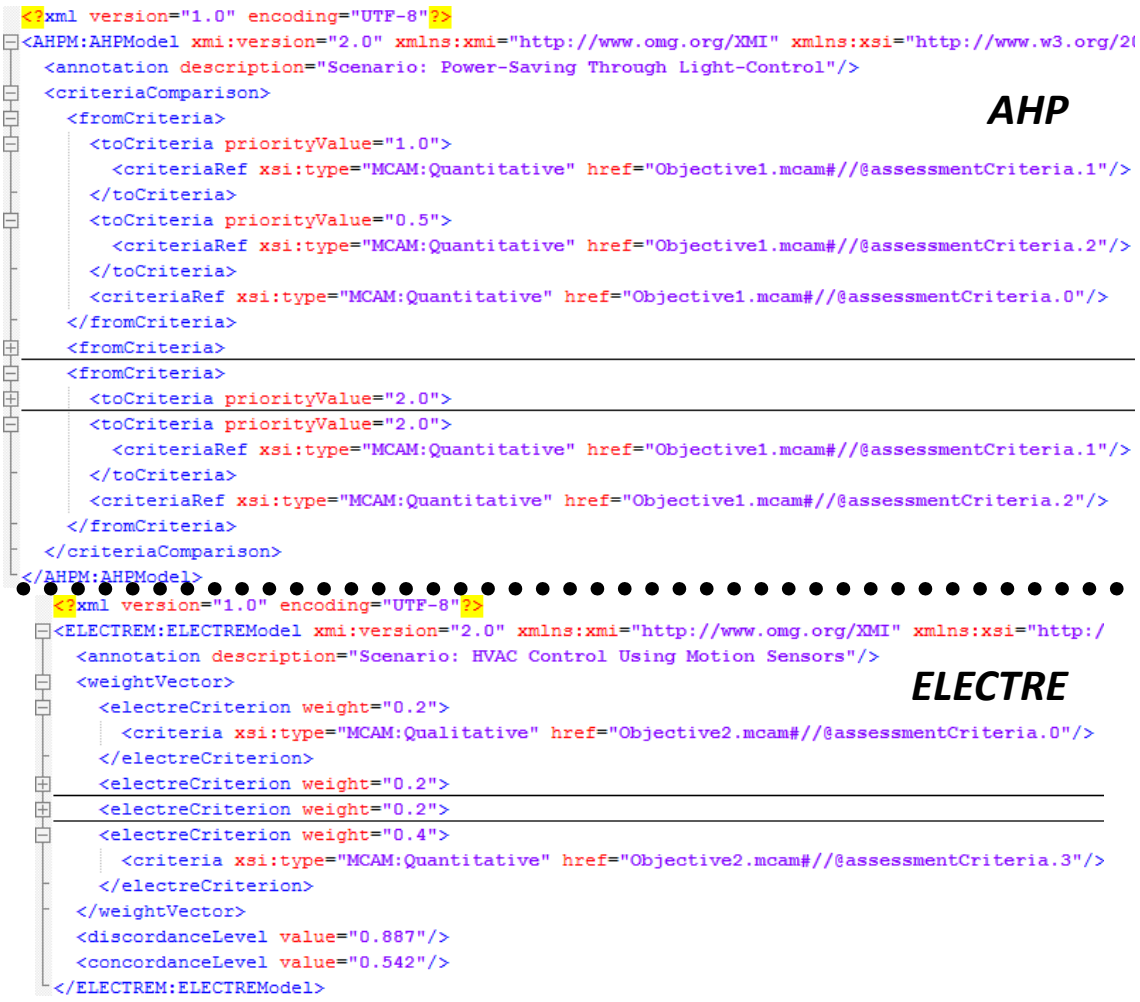


Figure 8.15: MCAM Methods: Instantiation for both Objectives.

From the figure is possible to identify the priority and weight values determined by stakeholders, defining the relation between criteria. Then each method has its own way

to evaluate solutions and consequently assess which is the more suitable solution (see Section 4.2). From the AHP model is concluded that criterion "*Light-Sensing Unit*" has the same importance than the second criterion, "*PWM Output Signal*", and half of the importance of criterion "*Price*". Priority values for criterion "*PWM Output Signal*" are hidden for document size reasons, but are the same as for criterion "*Light-Sensing Unit*". The last criterion, "*Price*", consequently presents twice the importance than the previous two.

Looking to ELECTRE model, shown on the bottom of Figure 8.15, stakeholders decided that the most important criterion is "*Cost*" with a weight of 40% ($weight = 0.40$) on the assessment, while the other three criteria ("*IEEE 802.15.4 Protocol*", "*Digital Output Signal*" and "*Motion-Sensing Unit*") have a 20 % weight ($weight = 0.20$) each. Discordance and concordance levels were computed following the ELECTRE methodology.

8.2.2 Implementation of Scenario 2: SensorML Standard

Standards have been proposed to define systems, modelling requirements, behaviours, processes, etc., supporting interoperability at a syntactic as semantic level. An example is SensorML [21], capable of represent physical as non-physical components as processes, for example detectors, actuators, mathematical operations or functions. SensorML is focus on describing sensor systems through a process, operation model to cover specifications, input/output, meta-data, etc., making a sensor system discoverable, measurable and observable. SensorML defines processes and processing components rather than give thorough description of its hardware.

The integration, interconnect of the proposed framework with SensorML standard evidence a clear benefit to the design of IoT Ecosystems. From one side, there is a framework capable of assist during the design phase of IoT Systems, with a complete and conscious form of represent IoT Systems and assess which is the more suitable to perform a certain task. Tasks which can be described by SensorML standard as all its operational aspects.

Therefore, and accordingly to the scenario presented in Section 8.1.2 — "Internet of Things — Simple Sensor", next are described the implementations steps that enable the interoperability between the proposed framework and the mentioned standard, demonstrating one of the framework features — interoperable with other systems.

Considering the design phase of an IoT Ecosystem, where the Multi-Criteria Framework to Assist on the Design of IoT Systems is used, the next step is to start defining IoT Systems deployment. With the use of the proposed framework, all information needed to match from IoT Systems assessment to functional aspects and processes is available.

In this sense, Figure 8.16 depicts the process of going from IoT Systems description and assessment specification to SensorML specification, using as example the test-case defined in Section 8.1.2. The left side of the figure represents the proposed framework specification models, and the right side is referent to SensorML.

To accomplish an automatic generation of a partial, pre-fill the description of sensor

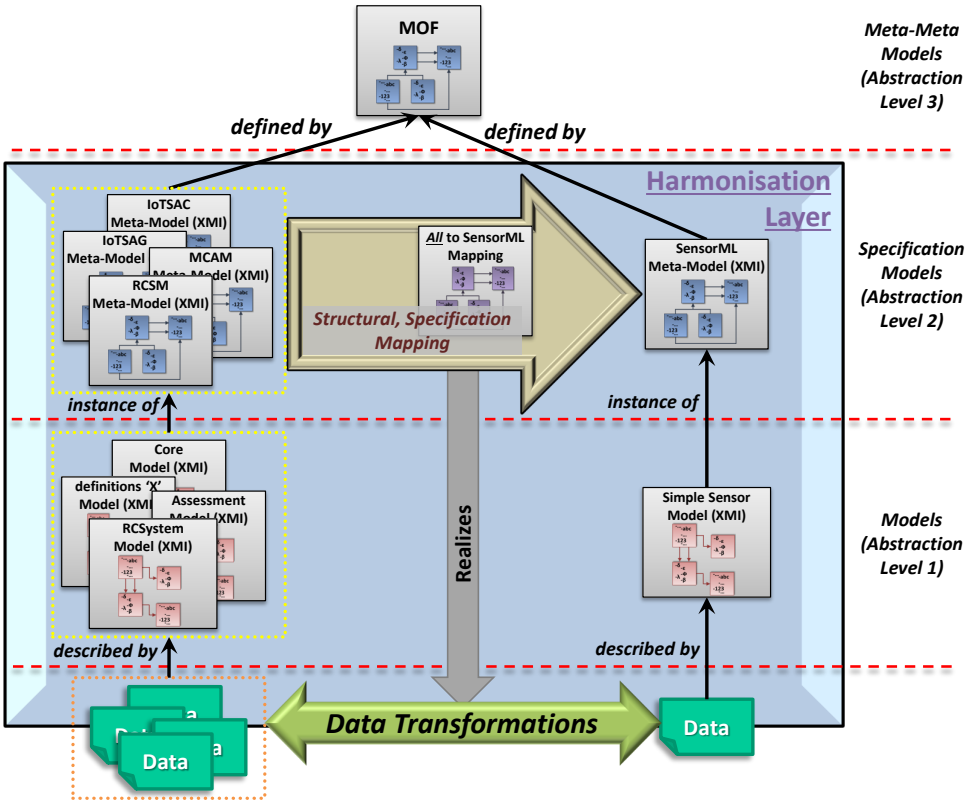


Figure 8.16: Harmonisation Layer: From Proposed Specifications to SensorML Specification (Internet of Things — Simple Sensor).

behaviour and processes (SensorML) a set of mapping rules must be specified. These rules are structural, specification mapping between the Meta-Models, performed at the second abstraction level accordingly with MDA. The resulting SensorML model is only partial; it is an initial formalisation, since the proposed work thus not addresses sensors, application actions or aspects regarding data exchange. The example, presented in Figure 8.8 shows that a SensorML model can contain the physical location of the sensor using GPS. The proposed specifications models are not able to describe the physical location, but can represent the GPS component itself.

Table 8.1 presents classes' and attributes matching between the two specifications. Where ***Source*** is referent to the specification models proposed in this work and ***Target*** is referent to the SensorML standard.

None of IoTSAC classes or attributes are mapped (directly) in this example, due to the fact that this example only describes a physical component processes. Although, being the core model, embracing all specifications, it is necessary to have it as ***Source*** to access to all other proposed specifications.

The sensor function description (***PhysicalComponent::description***) is obtained from the objective definition in attribute "*objective*" from class ***MultiCriteriaAnalysisModel*** of MCAM model. The objective for a multi-criteria assessment is defined in the beginning by the characterisation of the problem to solve.

Table 8.1: Harmonisation Layer: Structural, Specification Mapping to SensorML (Internet of Things — Simple Sensor).

Source Package:	Source Class / Attribute:	Target Class / Attribute:
IoTSAC	***IoTSystemsAnalysis***	*(not mapped)*
MCAM	***MultiCriteriaAnalysisModel***::*objective*	***PhysicalComponent***::*description*
	Rank::*rankingPosition*	***PhysicalComponent***::*id*
RCSM	***ResourceConstrainedSystem***::*name*	
	ResourceConstrainedSystem::*id*	***PhysicalComponent***::*identifier*
IoTSAG	***Property***; ***Unit***	***InputList***; ***OutputList***

To generate the ***PhysicalComponent*** in SensorML model, first is necessary to obtain the more suitable IoT System to execute the desired objective. Class ***Rank*** from MCAM contains the assessment result (numbered list) identifying each IoT System. Attribute *"id"* is then retrieved directly from the IoT System that finished in first place. In the defined SensorML test-case scenario the physical component has *"MY_SENSOR"* as *"id"* (see Figure 8.8).

The physical component identifier is a more specific, unique value to identify the component. In the proposed specification models this is obtained from attribute "id" of class ***ResourceContrainedSystem*** in RCSM. Attribute *"id"* derived from class ***SystemDescription*** of IoTSAG specification model.

The considered SensorML example, reports a simple temperature sensor providing online data observation. It does not contain any input information, commands but outputs the observed property — *temperature*. Property name, unit, domain are described in IoTSAG Meta-Model by classes ***Property*** (aggregation or single property), ***Unit*** and ***PropertyDomain***. At this moment, the property information is selected by stakeholders, however in the future a semantic analysis could be made to the objective text and retrieve the exact property(ies). Or at least give a smaller set of alternative properties.

8.3 Acceptance by Scientific Community & Industry

To be a contribution the research results must be published. Results should be published and shared with peers from the scientific community, and if appropriated transferred to the industry.

8.3.1 Acceptance by Scientific Community

Acceptance by the scientific community is important to verify the quality of the research work. To allow a continuous assessment of the work that has being developed, findings, contributions and research results must be publish, shared not only at the final but also

during the thesis elaboration. This enables the author to have an "on-line" verification of the work but others can continue the research or apply it into other areas.

During this thesis work, the author of this dissertation has published in a number of international conferences and scientific journals. Publications can be arranged accordingly with the groundwork themes that help to achieve the expected outcomes, as depicted in Figure 8.17.

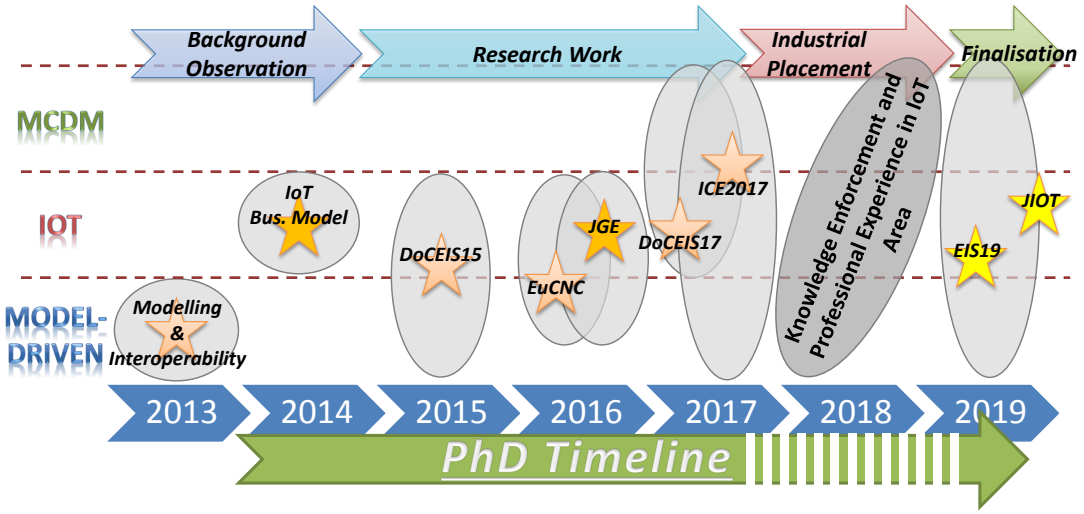


Figure 8.17: Acceptance by Scientific Community: Publications Timeline.

In a first phase, a background research was carried out and observations were made that lead to the identified issues that this dissertation addresses. In [198], the author published a work focus on information modelling and enterprises systems interoperability, which gave a perspective on a promising software engineering approach to simplify and formalise various systems activities and tasks (**Modelling & Interoperability**). IoT thematics, more exactly a business model for IoT Test-beds was published in [3]. During this research the author studied and faced numerous issues that needed to be addressed in IoT (**IoT Business Model**).

Identified the research area, a research question was drawn, "***How can Internet-of-Things Systems be designed to optimise the matching with the operating environment?***", and that give motto to the research work presented in this dissertation. In detail, the most important publications with inputs from this work are:

- **In Conferences:**
  - **DoCEIS15 [213]:** presented a model-based approach for Resource-Constrained Devices (RCS) energy test and simulation. It gave a first glimpse of a formal representation of an IoT System;
  - **EuCNC [214]:** focus on energy consumption awareness for Resource-Constrained Devices (RCS), by presenting an approach to direct measurement of consumed

energy, to streamline the process of building realistic models of power consumption;

- **DoCEIS17 [203]:** presented a multi-criteria analysis and decision methodology to analyse a set of different hardware platforms for the IoT;
- **ICE17 [32]:** presented a multi-criteria decision model to select a more suitable IoT System, with focus on an industrial scenario. This publication embraces the three areas used as background for thesis;

- **In Scientific Journals:**
  - **JGE [195]:** an extension of the work published in **EuCNC**, it include the study of a different development platform with focus on programmable logical, reconfigurable hardware platform — Field-Programmable Gate Array (FPGA);
  - **EIS19 [204]:** addresses a multi-criteria decision problem regarding the more suitable IoT System to perform a certain task for Cyber-Physical Systems (CPS). This publication embraces the three areas used as background for this dissertation.
  - **JIOT [215]:** Focus on presenting the overall thesis concept, an Assessment Methodology where is possible to apply Multi-Methods for Decision-Making, the use of Multi-Criteria and Multi-Constraints to analyse different solutions, all in benefit of a more conscious/aware IoT Ecosystems design. Submitted for review at the same time of the thesis document for defence (in 2019), it was accepted in July 2020 during the COVID19 pandemic and before thesis defence (September 2020).

8.3.2 Acceptance by Industry

Acceptance by Industry is of same importance as acceptance by peers in the scientific community. In case, results are not accepted by industry it is almost certain that the proposed contributions will never be used. Consequently, this research has been putted to test, to industrial validation in the metalworking sector. The prototyping scenario is based on the C2Net European Project, more specifically in a Portuguese pilot. The pilot, a Metalworking Small-Medium Enterprise (SME), aimed to improve the management of logistic flows and resources. The C2Net pilot is presented next in more detail.

8.3.2.1 C2Net's Metalworking Process Design

The C2Net Project goal was the creation of cloud-enabled tools to support supply network optimization of manufacturing and logistic assets based on collaborative demand, production and delivery plans. C2Net provided a scalable real-time architecture, platform and software that allow supply network partners to master complexity of a supply network to optimize the manufacturing assets by the collaborative computation of production plans.

C2Net's Metalworking Process Design focus mainly on two premises: provide more information to logistics so they can find best delivery prices and dates (among delivery/distribution services); and to improve delivery plans, giving a better response to customers' orders.

Scenario

The fabrication process starts when all necessary raw materials arrive at the factory for the fabrication of a metal post. The production depicted in Figure 8.18 starts by a bar code reader (**Start Process**), and finishes the same way, with a bar code reader (**End Process**). Along the process the raw material is cut (**Phase 1**), mould to desired shape (**Phase 2**) and painted (**Phase 3**).

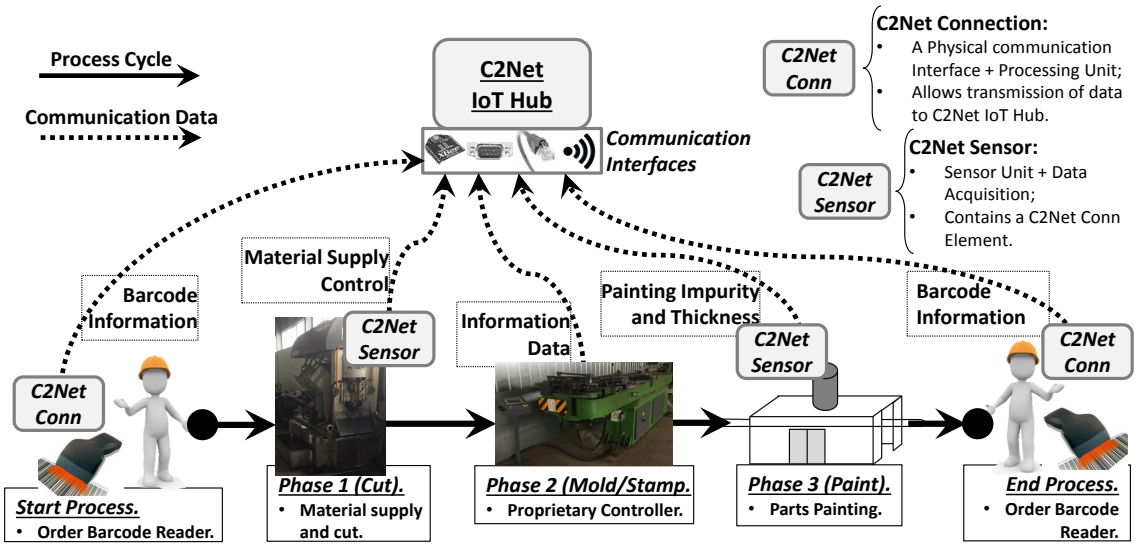


Figure 8.18: Industrial Scenario: C2Net's Metalworking Process Design.

Figure 8.18 shows the process steps, C2Net sensor functions (sensing functionalities) and C2Net IoT Hub that collects information from the fabrication process to optimise the supply network. The process stages are described next in detail:

- **Start Process:** operator starts by reading the order code using a bar-code reader. This bar-code reader has attached a C2Net Connection (C2Net Conn) module that is used to send the code information read to the central unit (C2Net IoT Hub);
- **Phase 1 (Cut):** is a step where the material is cut to a pre-defined size, from where each part will be created. Here, material supply and cut is controlled and this information is sent to the C2Net IoT Hub. The control focus on getting the relation between the supplied material and the amount of waste (number of pieces non-conforming);
- **Phase 2 (Mould/Stamp:** mould, gives the intended shape to each piece. The information from the stamp machine is managed by a proprietary controller and sent to

the central unit (C2Net IoT Hub). This machine, process phase does not need any additional computational module to share control information to system;

- **Phase 3 (Paint):** in this phase each part is painted. A C2Net Sensor is used to control the painting impurity and thickness of each painted part. This information is also sent to the central unit;

- **End Process:** is last step of the fabrication process. The operator reads again the order code using a bar-code reader to indicate that the process has ended. The code information is sent to C2Net IoT Hub using a C2Net Connection module.

The scenario, regarding IoT Systems, is composed by three types of components, modules: a C2Net IoT Hub; C2Net Sensors; and C2Net Connection (C2Net Conn) modules. A C2Net Sensor is composed by a sensor unit (e.g.: temperature, light sensor), a data acquisition module (e.g.: extra circuit elements needed for the sensor unit, signal amplifier), and a C2Net Conn module. A C2Net Conn module contains a physical communication interface (wired or wireless), a processing unit (e.g.: micro-controller) to handle the C2Net sensor instructions, functions or communication.

The C2Net IoT Hub selection, decision process will not be addressed in this scenario. It was implemented using a Raspberry PI version 2.0 with a Controller Area Network (CAN) module for communication, design (specifically to be used by the processing unit) and developed by author in collaboration UNINOVA-GRIS (Group for Research on Interoperability of Systems) integrated in the UNINOVA's CTS (Centre for Technology and Systems).

Objective Characterisation

The objective for the presented C2Net industrial scenario, the fabrication process of a metal post, is to select a proper, more suitable IoT System to act as C2Net Conn module. The decision making regarding the IoT Systems focus on C2Net Conn due to the fact that C2Net Sensors modules have different and specific sensing elements.

The objective has its characterisation divided in one pre-condition (wired communication) and five requisites (criteria) which are described next:

- **Wired Communication:** the pre-condition is related to the surrounding environment. In an industrial environment/factory, wireless communication can suffer great interferences due the amount of metal and iron. Therefore, a wired communication is advised. To reduce the complexity inherent to a fully-meshed network, and the delay in a point-to-point network, it is appreciated a bus communication. A bus communication network allows adding and removing C2Net Sensors, without any change to the physical network;

- **Energy:** the first requisite is related with energy consumption. Even with sensors connected to plant electric line (i.e. it not poses the problem of battery powered

devices, common to this type of sensors), energy consumption still is an aspect to consider because it will affect the factory electrical bill;

- **Implementation Time:** a straightforward solution is appreciated, and one aspects is the implementation time (e.g. purchase-delivery time of an item; code implementation time);
- **Implementation Difficulty:** to obtain a rapid deployment, difficulty (e.g. built/adapt hardware) is also an important aspect;
- **Cost:** from all the previous aspects, prerequisites it is impossible to forget the cost, always present in this type of operation/selection;
- **Processing Clock Speed:** is the last considered requisite. A good computation speed should also be taken into consideration.

With the identification of criteria, decision-makers (author, programmers, researchers, factory operators and owner) establish, set the priority values for each criteria. Table 8.2 presents the preference values between criterion, reflecting accordingly to decision-makers the importance, weight that each criterion has on the decision process.

Table 8.2: C2Net Scenario: Criteria Preference Relation.

Criteria:	Energy [milli-Watts]	Implem. Time [Days' Work]	Implem. Difficulty [Difficulty]	Cost [€]	Processing Clock Speed [MHz]
Energy	1	1/3	1/3	1/3	2
Implem. Time	3	1	1	1	4
Implem. Difficulty	3	1	1	1	4
Cost	3	1	1	1	4
Processing Clock Speed	1/2	1/4	1/4	1/4	1

Analysing the outcome of decision-maker's judgement is possible to state that criteria **Implementation Time**, **Implementation Difficulty** and **Cost** are the most important and present the same preference weight. Criterion **Energy** is less important than the previous three but is more important than criterion **Processing Clock Speed**.

As an example let's consider the first weight line from Table 8.2. **Energy** has weight 1 to its own, therefore equal importance. It is 3 times less important than the next three criteria (criterion 2, 3 and 4). But criterion **Energy** has a relation of 2 with last criterion (**Processing Clock Speed**), meaning that **Energy** it is more important than **Processing Clock Speed**.

Once more the decision-makers are call to dictate, define criteria constraints for the desired objective. The selected Constraints which were applied to the possible solutions (IoT Systems) are divided in two of the three possible types of constraints:

- **Optimization:** the first four criteria must be optimise by minimisation, and to (**Processing Clock Speed**), the last criterion, must be applied maximization;
- **Condition:** criterion **Implementation Time** is limiting the use of IoT Systems that take more than a 8 days' work to be implemented (or enables the use of solutions that take less than 8.1 days' work). IoT Systems are only considered if their criterion **Implementation Difficulty** presents values lower than "*hard*".

In this scenario the availability constraint was not applied.

As possible solutions, IoT Systems, to act as C2Net Connection module in the presented C2Net scenario, were identified six hypotheses. Each one is a combination of micro-controller unit with a Controller Area Network (CAN) communication module. The six IoT Systems (the solutions) considered are:

1. s_1: an Arduino UNO R3 with a CAN Shield, both bought (off-the-shelf). This IoT System is identified as "***UnoRev3 + CANShield***";
2. s_2: an Arduino UNO R3 (bought — off-the-shelf) with a new full CAN board (new design and construction). This IoT System is identified as "***UnoRev3 + CompleteCANcomm***";
3. s_3: an Arduino DUE with a CAN Shield, both bought (off-the-shelf). IoT System identified as "***DUE + CANShield***";
4. s_4: an Arduino DUE (bought — off-the-shelf) with a new built CAN board (new design and construction). Identified as "***DUE + CANcomm***";
5. s_5: a micro-controller board (new design and construction) and a CAN Shield (bought — off-the-shelf). This IoT System is identified as "***ATMEGA328P + CANShield***";
6. s_6: a micro-controller board with a new full CAN board (both from a new design and construction). IoT System identified as "***ATMEGA328P + CompleteCANcomm***".

The first four are based on two types of Arduino hardware development platforms, varying the CAN module used. The last two IoT Systems follow mainly a built from scratch approach. Figure 8.19 shows graphically the arrangement between the different micro-controllers and CAN boards. s_i identifies the solution i, a combination of a micro-controller with a CAN module.

To give a better notion of the criteria values, their differences, and to clarify the imposed IoT Systems concurrency by assessment criteria, it is presented in Table 8.3 the

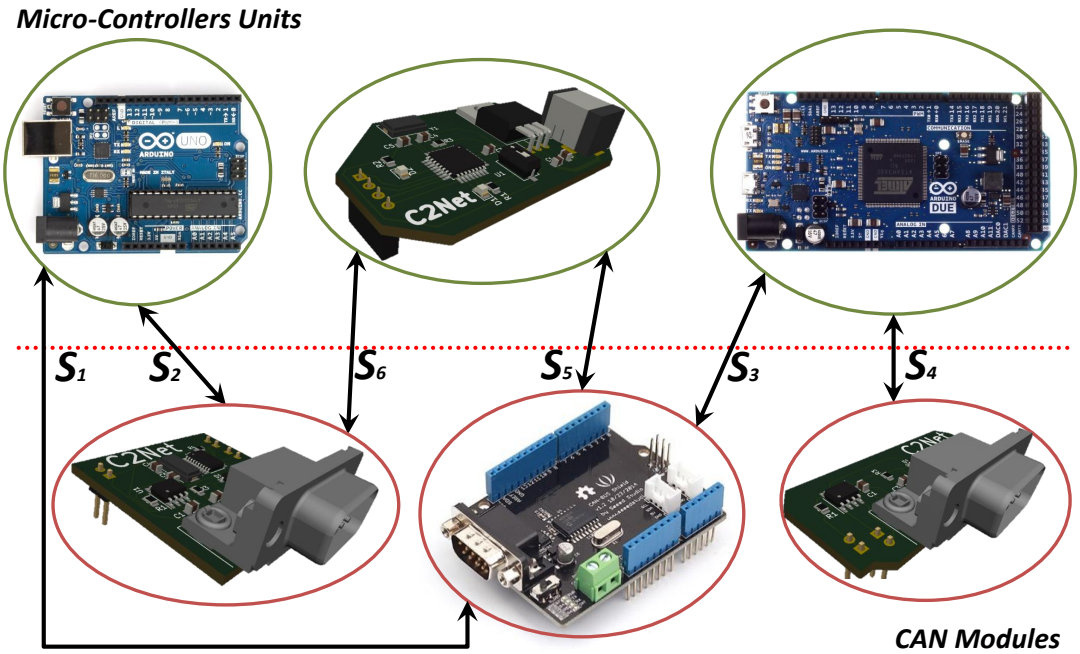


Figure 8.19: Industrial Scenario: Considered IoT Systems (Hardware part).

data values for each criterion. Figure 8.20 presents the same data in a radar chart. This small data set is a clear example on how difficult a decision can be for decision makers, if there is not a tool to assist them in their choice.

Table 8.3: C2Net Scenario: Criteria Values.

Criteria:	Energy [milli-Watts]	Implem. Time [Days' Work]	Implem. Difficulty [Difficulty]	Cost [€]	Processing Clock Speed [MHz]
UnoRev3 + CANShield	127.5	1	1	46.2	16
UnoRev3 + Complete-CANcomm	127.5	5	4	30.2	16
DUE + CAN-Shield	372.9	1	1	61.87	84
DUE + CAN-comm	372.9	5	3	43.57	84
ATMEGA328P + CANShield	100.7	5	3	28.1	20
ATMEGA328P + Complete-CANcomm	100.7	10	5	12.1	20

It is important to state that the presented values for criterion energy, are in fact energy consumption values from the operating system footprint. Each IoT System was programmed with the application code and real tests were made to measure the energy

consumption. Tests were made outside the C2Net network.

Figure 8.20 is a radar graph with criteria values from each IoT System. Values are normalised to facilitate the graph interpretation, since the difference between the ***Energy*** criterion and the others is high. IoT Systems are identified as s_i and the criteria as c_j accordingly with Equation 8.3 and 8.4, respectively.

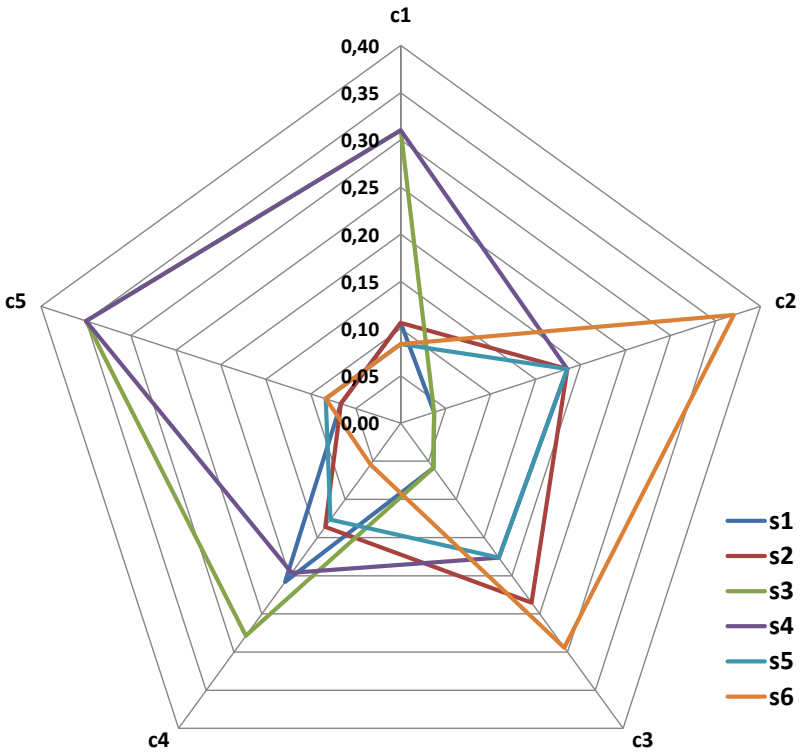


Figure 8.20: Industrial Scenario: Criteria Values (Radar Chart).

Assessment Methodology

With the objective defined, criteria, respective constraints, priorities and the possible solutions (IoT Systems) to act as C2Net Connection modules was possible to apply the multi-criteria assessment methodology presented in Section 6.5.

In this sense, and following the proposed methodology, the ***Solutions*** represented by set S_{Set} and ***Criteria*** represented by set C_{Set} are defined in Equation 8.3 and 8.4, respectively. Where $n = 6$ and $m = 5$.

$$S_{Set} = \{UnoRev3 + CANShield, UnoRev3 + CompleteCANcomm, \\ DUE + CANShield, DUE + CANcomm, ATMEGA328P + CANShield, \\ ATMEGA328P + CompleteCANcomm\} \tag{8.3}$$

$$C_{Set} = \{Energy, ImplementationTime, ImplementationDifficulty, \\ Cost, ProcessingClockSpeed\} \tag{8.4}$$

The solutions set, S_{Set}, was obtained, created from the IoTSAC file, by going through the list of *ResourceConstrainedSystem* classes from RCSM file. The list of criteria, C_{Set}, is given by MCAM model.

With the sets of possible solutions and criteria defined is now possible to create ***Assessment Table***, A_{table}, presented in Equation 8.5 that describes the solutions-criteria cluster.

$$A_{table} = \begin{bmatrix} 127.5 & 1 & Easy & 46.2 & 16 \\ 127.5 & 5 & Medium-Hard & 30.2 & 16 \\ 372.9 & 1 & Easy & 61.87 & 84 \\ 372.9 & 5 & Medium & 43.57 & 84 \\ 100 & 5 & Medium & 28.1 & 20 \\ 100 & 10 & Hard & 12.1 & 20 \end{bmatrix} = (v_{i,j}) \in \mathbb{R}^{6\times5} \tag{8.5}$$

The list of criteria applied to each criterion is depicted in Equation 8.6, where each line represents jth criterion. For example, the first line represents the criterion **Energy**, $j = 1$.

$$\begin{gathered} Constraints = \{MIN; \\ MIN, LessThan; \\ MIN, LessThan; \\ MIN; \\ MAX;\} \end{gathered} \tag{8.6}$$

Applying the *Optimization* constraints to the solutions-criteria cluster, A_{table}, using the Assessment Constraints function, results in $OptA_{table}$ shown in Equation 8.7. This is an intermediate matrix that represents solutions-criteria values changed by the influence of *Optimization* constraints.

$$OptA_{table} = \begin{bmatrix} 245.4 & 9 & 4(Medium-Hard) & 15.67 & 16 \\ 245.4 & 5 & 1(Easy) & 31.67 & 16 \\ 0 & 9 & 4(Medium-Hard) & 0 & 84 \\ 0 & 5 & 2(Easy-Medium) & 18.3 & 84 \\ 272.9 & 5 & 2(Easy-Medium) & 33.77 & 20 \\ 272.9 & 0 & 0(Extra-Easy) & 49.77 & 20 \end{bmatrix} = (v_{i,j}) \in \mathbb{R}^{6\times5} \tag{8.7}$$

The values for criterion **Implementation Difficulty** were converted from their qualitative values to a quantitative scale. This mapping is presented in Table 8.4.

Table 8.4: C2Net Scenario: Conversion of criterion "Implementation Difficulty".

Extra-Easy	Easy	Easy-Medium	Medium	Medium-Hard	Hard
0	1	2	3	4	5

The proposed framework enables the use of any method to rank the solutions. More precisely, Section 6.5 — ***IoT Systems: Multi-Criteria Assessment Methodology***, described how methods can be integrated with the assessment methodology. The selected MCDM method for this C2Net scenario was the Analytic Hierarchy Process (AHP). Therefore applying MCDM Method Procedure with AHP, results on the output, $MCDM_{RO}$, given by Equation 8.8.

$$MCDM_{RO} = \begin{bmatrix} 0.2189 \\ 0.1508 \\ 0.1840 \\ 0.1416 \\ 0.1799 \\ 0.1248 \end{bmatrix} = (RO_{i,1}) \in \mathbb{R}^{6\times 1} \tag{8.8}$$

The element $RO_{i,1}$ presents the AHP method rank outcome value for IoT System ***i***.

Computed the result from the AHP method was then applied the ***eliminative assessment constraints***, that in this case consists only of ***Condition*** constraint type. The enforcement of assessment constraints of type ***Condition***, AC_{Cod}, results in a 5-by-1 binary matrix, $Eval_{Cod}$, given by Equation 8.9.

$$Eval_{Cod} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = (Eval_{Cod})_{i,1} \in \mathbb{N}^{6\times 1} \tag{8.9}$$

The ***Th*** argument is the threshold value for constraint ***k*** in criterion ***j***, value that is retrieved from the instantiation of MCAM. Threshold value, ***Th***, for criterion ***Implementation Time*** is 8.1 and for criterion ***Implementation Difficulty*** is "hard". The element $(Eval_{Cod})_{i,1}$ represents the outcome for IoT System ***i*** by applying all constraints of the ***Condition*** type.

Since there is no constraints of type Availability to be applied the ***final solutions ranking***, $(IoT_{Systems})_{Rank}$, is given by Equation 8.10. Where $(S_{Value}^{Outcome})_{i,1}$ is the Multi-Criteria Assessment outcome value for the solution ***i*** (IoT System ***i***).

$$\begin{aligned} (IoT_{Systems})_{Rank} &= MCDM_{RO} \odot Eval_{Cod} \\ &= \begin{bmatrix} 0.2189 \\ 0.1508 \\ 0.1840 \\ 0.1416 \\ 0.1799 \\ 0.0 \end{bmatrix} = ((S_{Value}^{Outcome})_{i,1}) \in \mathbb{R}^{6\times 1} \end{aligned} \tag{8.10}$$

Sorting the ***final solutions ranking*** list it is obtained the multi-criteria assessment of all six possible chosen solutions (IoT Systems) ordered from the more suitable to the less suitable IoT System to perform the designated task. For the presented scenario (see Figure 8.18), the fabrication process of a metal post, the proper IoT System to act as C2Net Connection (C2Net Conn) element is the Arduino Uno R3 with a CAN Shield. The complete and ordered result is presented in Table 8.5.

Table 8.5: C2Net Scenario: IoT Systems Final Ordered Ranking.

Position:	**IoT System:**	**Outcome Value:**
1	UnoRev3 + CANShield	0.2189
2	DUE + CANShield	0.1840
3	ATMEGA328P + CANShield	0.1799
4	UnoRev3 + CompleteCANcomm	0.1508
5	DUE + CANcomm	0.1416
6	ATMEGA328P + CompleteCANcomm	0

This study, perform during the IoT deployment design phase for the factory ground floor (in this case an industrial scenario — C2Net Portuguese pilot), assisted stakeholders on the decision of which was the more suitable IoT System to act as the C2Net Connection module. Stakeholders were able to perform a more conscious decision taking into consideration different and in some cases contradictory factors. Notwithstanding the fact that the number of elements analysed was small, and even so the assessment methodology proved to be very useful. The considered solutions were based on a single hardware platform (Arduino) that lead also to a single programming language, and it was only considered a short list of criteria and constraints. In bigger IoT Ecosystems the proposed framework helpful impact will be even higher.

8.3.2.2 Envisaged Integration with vf-OS

Virtual Factory Open Operating System (vf-OS) is a co-funded European Project under H2020 and is already in its final year. Composed by 14 partners (Users, Technology Providers, Consultants and Research Institutes) of 7 different countries, vf-OS has the objective of develop an Open Operating System for Virtual Factories. Aiming in providing a Software Development Kit (OAK) to be used by software developers to build and deploy Manufacturing Smart Applications for industrial users. Applications stored in the Virtual Factory Platform, a economical multi-sided market platform, enables the value creation from customers groups interactions (Software Developers, Manufacturing and Logistics Users, Manufacturing and Logistics Solution Providers, and Service Providers) [216].

vf-OS focus on provide a range of services for Factories of the Future integrating better manufacturing and logistics processes, to become a reference software managing factory' components system. Factory components are hardware and software resources, on which

vf-OS provided common services for factory computational programs. A high-level view of vf-OS Architecture is presented in Figure 8.21, retrieved from [217].

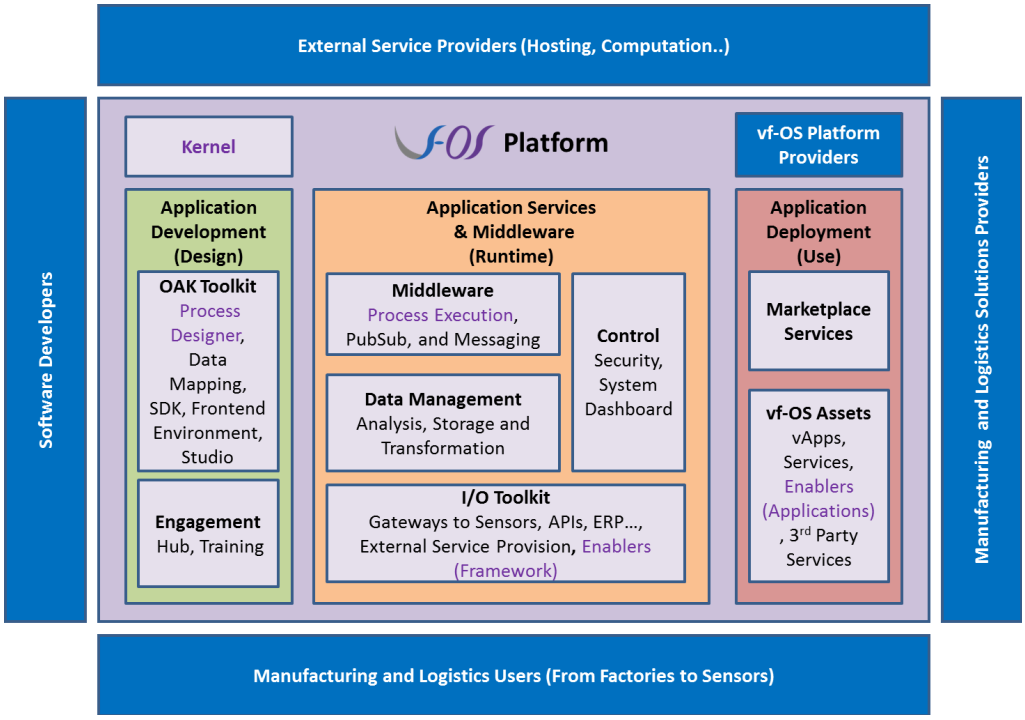


Figure 8.21: High-Level View of vf-OS Architecture (retrieved from [217]).

Clearly the author's proposed work hooks up with "**Application Development**" architectural building block, for its use within the design phase. This block embraces different vf-OS components that assist developers, analysts on the development of assets/applications.

The vf-OAK Software Development Kit (SDK) is a central environment for applications development, providing centralised access to vf-OS assets and functionalities. It supplies components, resources and services to other vf-OS architecture parts (e.g.: Studio) as they required. To this moment are provided 30 SDK User's Story (SDUS) for assets search, invoke, deploy, and store services. Identifying some assets and functionalities provided by the vf-OAK SDK, and considered to be related to this work, are found ***SDUS003 — Get Data Analytic Services***; ***SDUS004 — Get Enablers***; and ***SDUS006 – Get Drivers*** [216].

Each one of these tasks is described next, converging, focusing on these work contributions:

- ***SDUS003 — Get Data Analytic Services:*** list existing Data Analytic services. It is focused on analyse events from stream and historic process data within the manufacturing domain;

- ***SDUS004 — Get Enablers:*** list existing vf-OS Enablers. Enablers are responsible for

expose their service interfaces, identifying the needs to understand and implement the services diverse functionalities;

- ***SDUS006 – Get Drivers:*** list existing Drivers. vf-OS Drivers provides a set of functionalities such as register physical devices in vf-OS and specification of their parameters, devices' data reading or controlling devices.

It is envisaged the integration of Multi-Criteria Assessment of IoT Systems as an additional Analytic Service (accessible through ***SDUS003***), enabling developers to consciously, properly assess which is the more suitable IoT System(s) for a certain task. The formal specifications proposed in this work can also contribute to a more straightforward, better description of IoT Systems parameters, therefore a useful vf-OS Driver (accessible through ***SDUS006***). Since this poses as an integration of new services, it will be needed new vf-OS Enablers to expose these new services interfaces (accessible through ***SDUS004***).

8.4 Hypothesis Validation

Section 1.4.2 presented the considered hypothesis for this dissertation, derived from the background analysis and stated research questions. In order to remember and verify the assumption, it is included here:

> *"If in an Internet-of-Things system design, engineers could use a multi-criteria framework to simulate and analyse hardware and software solutions, then proper solutions could be selected and applied, leading to a more suitable design of an Internet-of-Things System."*

The outcomes observation, collected from implementations and proposed methods testing, drives the author to conclude that the hypothesis has been validated. In fact, it was demonstrated that the multi-criteria assessment framework is capable to perform an analysis of different solutions, assisting stakeholders during the design phase of IoT Deployment. It takes into consideration different and even contradictory aspects that have a direct impact not only on IoT Systems performance but also on overall ecosystem performance. Also, the description of IoT Systems using formal specifications, to describe its hardware components and enabling the specification of different software languages, presented as a powerful mechanisms to specific assessment criteria but it is also to the use, integration, or improvement of other tools or systems (such as energy simulation tools or for an IoT higher-level description systems).

These work contributions have been recognised and accept by scientific experts through publications acceptance and collaboration in international projects. It was also possible to validate this work on industrial scenarios, showing a complete external consensus and acceptance regarding the research work developed.

Chapter 9

Conclusions

This chapter provides a summary of the research developed, how it was conducted and identifies the main contributions. First, an overview of the path taken, identifying research activities that had somehow an important role on this research work. Followed by a description of the main contributions, highlighting the publications made during this PhD thesis and also mentioned others that are not directly related with the research work. To finalise it is proposed some future research.

9.1 The Path from Background Research up to PhD Thesis

This PhD work was developed based on built experience from involvement, and knowledge acquired in distinct European projects, working with differentiated research teams. At an early stage, the author had the opportunity to study two important themes used as background thematics in this dissertation. Figure 9.1 depicts the author progress and involvement with other research activities, leading to this PhD thesis. The figure presents along four stages, the background thematics cross-referencing with European Projects that had more relevance to this work theme.

The first phase, identified as ***Background Observation*** in Figure 9.1, allowed knowledge enrichment about IoT and model-based techniques and methods, as well as the problem identification that this work addresses.

Identified the scope and problem to address, the author carry on the background study and started the research work development, as identified in the figure as ***Research Work***. The work conclusion was carried out in two phases: the first (identified in figure as ***Industrial Placement***) consisted on obtaining professional experience in the PhD area and verify the work developed; the second phase (identified in figure as ***Finalisation***) consisted on the doctoral program finalization by writing this document. The following

subsections will describe each phase overtaken, and how did it influence this PhD work. By the help provided on improving the know-how regarding this thesis scope or in the enforcement of the developed work.

9.1.1 Background Observation

During the ***Background Observation*** phase, identified in Figure 9.1, the author had the opportunity to get actively involved in European Projects and work with other research teams, exchanging personal experience and knowledge, improving in this way the know-how regarding this thesis scope. Next is describe each European Project, and how it contributed to the author work.

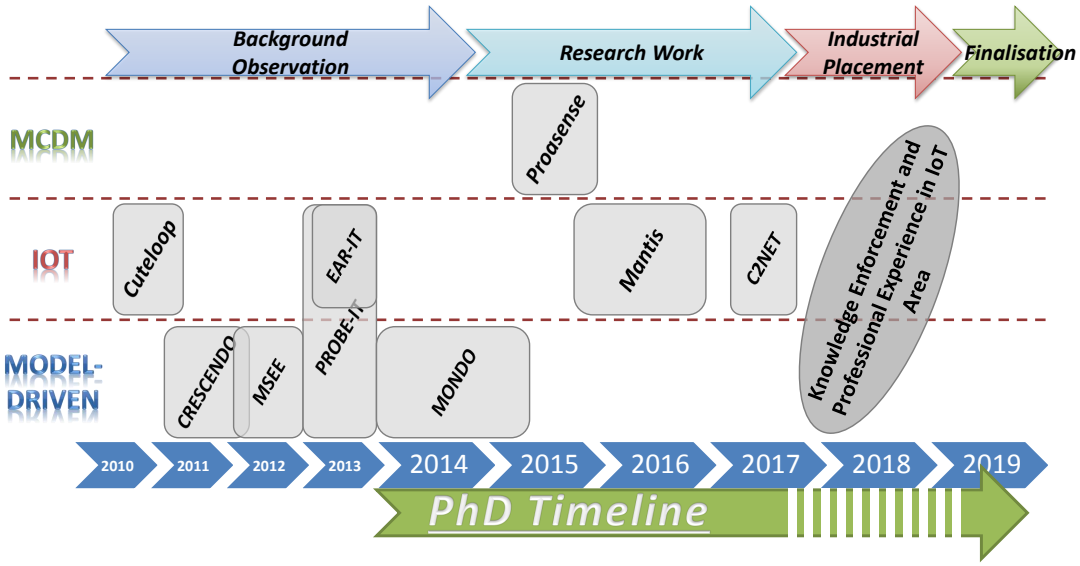


Figure 9.1: The Path from Background Research up to PhD Thesis.

9.1.1.1 Cuteloop and EAR-IT

"Customer in the loop: using networked devices enabled intelligence for proactive customers integration of the drivers integrated enterprise" (Cuteloop) was an European Project co-funded, launched under Seventh Framework Programme (FP7) that had bring together nine organisations from six different countries. The project focus was to create a new approach for distributed and asynchronous control of business processes, employing "Networked Devices Enabled Intelligence", bringing diverse actors (particularly customers) to interact in an integrated enterprise scenario.

"Experimenting Acoustics in Real environments using Innovative Test-beds" (EAR-IT) was also an European Project co-funded, launched under Seventh Framework Programme (FP7), coordinated by UNINOVA — Instituto de Desenvolvimento de Novas Tecnologias, a team that the author was part of. This project had bring together seven organisations all from different countries. The project aimed to validate and confirm the Research and

Technological Development (RTD) possibilities of using audio data, both in indoor as in outdoor environments. With validation on large-scale and innovative test-beds that support experimental research, namely the SmartSantander and HobNet — the FIRE facilities.

Both these projects give a deep insight over varied IoT thematics, with the objective of deliver new innovative range of services and applications to many IoT areas, namely Industry 4.0, smart-buildings and smart-cities. Besides a high level of practical experience, these projects also allow the author to clearly identify the issues that this work addresses. In other words, manufacturers are engaged in developing new embedded systems for different purposes, addressing a wide set of application domains and services, unlocking a variety of hardware and software solutions, without a clear way to formally describe IoT Systems, capable of being used by applications in an automatic way. Cuteloop in [194] presented a sensor node architecture using graphical representation which was used as background for the proposed IoT System hardware formal specification (see Section 5.4.1).

9.1.1.2 CRESCENDO, MSEE and MONDO

"Collaborative & Robust Engineering using Simulation Capability Enabling Next Design Optimisation" (CRESCENDO), an European Project co-funded, launched under FP7, with 59 partners involved from 13 different countries, including major aeronautics industry companies (e.g.: Airbus), service and IT solution providers (e.g.: Siemens), research centres and academic institutions. It was a research and technology project led by Airbus, with the ambition to make a step change in the way that modelling and simulation activities are carried out, by multi-disciplinary teams working as part of a collaborative enterprise, in order to develop new aeronautical products in a more cost and time efficient manner. It provided a generic Business Object Model, web services and Data Exchange (DEX) specifications built on ISO standards, supporting collaborative simulation built upon flexible workflows, traceability, re-usability, and advance interoperability.

"Manufacturing Service Ecosystem" (MSEE), also an European Project co-funded, launched under FP7, with 22 participants from 8 different countries, focus on transform current manufacturing hierarchical supply chains into manufacturing open ecosystems. This project aimed to create new Virtual Factory Industrial Models, where service orientation and collaborative innovation can support a new renaissance of Europe in the global manufacturing context.

"Scalable Modelling and Model Management on the Cloud" (MONDO), a Specific Target REsearch Project (STREP) of FP7, focus on tackle the increasingly important challenge of scalability in MDE in a comprehensive manner. With 9 partners involved in achieving scalability in modelling and MDE, by enabling teams of modellers to construct and refine large models in a collaborative manner, advancing the state-of-the-art in model querying and transformations tools so that they can cope with large models (of the scale of millions

of model elements), and providing an infrastructure for efficient storage, indexing and retrieval of large models.

CRESCENDO, MSEE and MONDO where three projects that enabled the author to get acquaintance with problems such as information exchange between different companies, sustainable interoperability, and scalability. But these projects also present methodologies, methods to address such issues like using model-driven techniques capable of modelling systems at different levels of abstraction. Model-driven approaches were presented in detail in Section 3, and applied to the proposed framework, enabling formalisation of all contributions, presenting methodologies to an interoperable framework with any kind of system.

9.1.1.3 PROBE-IT

"Pursuing ROadmaps and BEnchmarks for the Internet of Things" (PROBE-IT), a co-funded European Project launch under FP7 with 9 participants from 7 different countries with the particularity on involving 4 different continents (Europe, Asia, Africa and South America). The project aimed to complement the global portfolio with benchmarks, roadmap and other key inputs on validation and interoperability. It was also focus on providing overall support, "Hitchhiker's" guide, to current and future IoT research programmes addressing all these important and technical issues.

This project addressed transversely the IoT scope, providing, among other themes, prospective views on aspects such as: IoT Business Models; IoT Interoperability & Standards; and IoT Human Factors with focus on Privacy. Focus on giving answers regarding where the IoT is heading, in its many dimensions; what technologies/issues to consider for IoT deployment in the future.

9.1.2 Research Work

The second period identified in Figure 9.1, ***Research Work***, corresponds to the beginning of contributions development with a constant, continuous background study. Insights over multi-criteria decision thematics, were fortified with participation and collaboration with research teams and contributions were validated through industrial scenarios.

9.1.2.1 Proasense

"The Proactive Sensing Enterprise" (Proasense), was an European Project co-funded launch under FP7 with 8 participants from 6 different countries, where it is included two large industrial partners, one Small-Medium Enterprise and five research organisations. The project goal was to aid a class of enterprise systems — proactive enterprises, to be continuously aware of what "might happen" in the relevant business context and optimize their behaviour to achieve that what "should be the best action". This was proposed to be achieved by an efficient transmission from Sensing into Proactive enterprises.

This project gave the opportunity to the author to be involved in the design and develop methods for supporting enterprise decision making. The creation of specification models to define Key Performance Indicators (KPI) with contextualized target values, with proactive validation through offline and online KPI values analysis, and the creation of a proactive-based Business Models.

9.1.2.2 Mantis

Cyber Physical System based Proactive Collaborative Maintenance (Mantis), was a co-funded, H2020 European Project with 50 partners from 12 different countries. Its goal was to develop a proactive maintenance service platform architecture based on Cyber-Physical Systems (CPS) that allows estimation of future performance, predict and prevent imminent failures, and schedule proactive maintenance.

Working directly with a Portuguese company — Adira Metal Forming Solutions S.A. (leading manufacturer and global supplier of sheet metal working machinery, specializing in laser cutting, hydraulic press brakes, shears, robotised bending), the author had the opportunity to apply and improve the know-how regarding IoT and interoperability themes. The author was involved in prerequisites definition, architecture design taking into account interoperability issues. Also, analysis of already available machine sensors; study of market and research sensors suitable to the use case; study and analysis of hardware components suitable for the design of new sensor boards based on features such as sound, infra-red and motion.

9.1.2.3 C2Net

Cloud Collaborative Manufacturing Networks (C2NET), a co-funded European Project launch under H2020 with 20 participants from 6 different countries with the objective of create cloud-enabled tools, based on a scalable real-time architecture, enabling supply network optimization of manufacturing and logistic assets, taking into consideration collaborative demand, production and delivery plans. Decision makers can manage information, process and store with a complete visibility and real-time status of the entire supply chain with capability to control, share and collaborate.

This collaborative project adds specific contributions from this PhD, namely a more conscious, aware selection of IoT Systems for a use-case scenario. Contributions were applied to improve the management of logistic flows and resources of a Portuguese metalworking Factory (Antonio Abreu Metalomecanica, LDA.). This scenario enabled the use of the formal specifications as well as the use of the multi-criteria assessment of IoT Systems, to assess the overall performance of each solution according to the sensing needs of each machine. The validation of the proposed framework, using the C2Net Portuguese pilot was presented in Section 8.3.2.

9.2 Scientific and Technical Contributions

This research work' vision (see Section 1.2) foreseen the proposal of a framework capable of fully characterise an IoT System and assist stakeholder's to perform a more conscious, aware, accurate decision regarding which is the proper IoT System for a specific task. The framework also intended to provide methods for energy-simulation tools integration and address interoperability with standards, methods or systems within the IoT scope.

The presented theme, although ambitious, is an important research challenge, proposing relevant scientific contributions to the research community and consequently complements the State-of-the-art.

Scientific and technical contributions addressed the lack of formal descriptions of IoT Systems, in a highly diversified market where manufacturers are providing numerous devices, proposes means to assist stakeholders in improving their decisions quality, rather than continue to believe in lived experience or intuition, in a wide set of features/criteria that influence the overall performance of an IoT System. Literature review in Section 2.3, that only scraped standards topic in IoT scope, showed a numerous variety of possible solutions in many IoT areas (e.g.: data, communication protocols, platforms, OSs, security), with a broad number of features to analyse. With a model-driven nature framework and well-defined formalisations coexistence with other tools, systems, standards or methods was also addressed.

Next is presented this research work outcomes in terms of scientific and technical contributions, industry placement and accomplished publications.

9.2.1 From a Research Question to Validation

Section 1.4 presented the research question that this PhD Thesis addresses. In order to remember and verify the question, it is included here:

> *"How can Internet-of-Things Systems be designed to optimise the matching with the operating environment?"*

This question was decomposed in three sub-questions, identified by the author as key to solve the main research question, ensuring in this way the research focus and targeted results. Table 9.1 presents the main outcomes derived from this research work contributions, split over the research sub-questions.

The presented outcomes clearly addressed the question of how to optimise the matching of IoT Systems with the operating environment, by proposing means to improve the design of IoT Systems. However, literature has been focus on functional/behaviour, activities/actions/interactions within an IoT Ecosystem, which cannot be left apart. In this sense, and to not isolate the proposed work and in somehow fill the gap between design (addressed in this work) and management phases of an IoT Ecosystem, mechanisms were

Table 9.1: Relation Between Research Sub-Questions and Contributions.

Research Sub-Question:	**Q1.1:** "Which methods could be applied or develop to formally describe an IoT System (hardware, software, energy)?"
Contributions and Outcomes:	A framework that formally describes a complete IoT System (Chapter 5). It is mandatory to have a hardware and software characterisation, but if available it can also include an energy profile: • A single specification model, Resource-Constrained System Hardware (RCSH), to describe any IoT System hardware (see Section 5.4.1); • Different software languages to describe IoT System application code. Domain experts can include their own programming language. Conceptual approach in Section 5.4.2, while Section 8.1.1.2 complements it with two software languages examples; • Section 5.4.3 proposes an approach to in the future an energy analysis can be achieved based on specific IoT Systems descriptions.
Research Sub-Question:	**Q1.2:** "Which methods could be applied or develop to assist in IoT System assessment?"
Contributions and Outcomes:	An IoT Systems Multi-Criteria Assessment Methodology (Section 6.5), based on a specification model, Multi-Criteria Analysis Meta-Model (MCAM), presented in Section 6.3, to: • Enable stakeholders to assess which IoT System is more suitable to their application scenario and purpose; • A concrete form to describe criteria than influence the overall performance, define constraints or restrictions for each criterion; • Use of different (even new or user-defined) MCDM methods, with the possibility of being applied within the same problem; • Two specification models (Section 6.4) are presented to describe AHP and ELECTRE MCDM methods.
Research Sub-Question:	**Q1.3:** "Which multi-criteria decision framework would provide a suitable decision support for the design of IoT Systems?"
Contributions and Outcomes:	It is proposed a novel model-driven multi-criteria assessment framework to analyse IoT Systems, capable of suggesting the more suitable IoT System to execute a certain task (Chapter 6): • Criteria are based on IoT Systems formal descriptions (see outcomes of Q1.1); • Applies a novel multi-criteria assessment methodology to improving assertiveness regarding IoT Systems selection (see outcomes of Q1.2).

proposed to achieve an automatic coupling (see Sections 7.4 and 8.2.2) with standards that define systems interaction, modelling requirements, behaviours, processes, etc.

9.2.2 Publications Summary

Along the research work development several publications were made in conferences and journals. The next two sub-sections will present respectively the publications made in

conferences and the ones made in journals. It will also be mentioned publications that are not directly related with this work.

9.2.2.1 Publications in Conferences

Table 9.2 shows the accomplish conference publications. The first part of the table identifies publications directly related to the envisaged work, followed by publications, "Other Publications", accomplished by parallel research. Publications are organised by year (first column), and it is highlighted publication, conference and/or book name.

Table 9.2: Accomplished Publications in Conferences.

Year:	Title:	Conference/Book:	Ref:
2013	Achieving Interoperability via Model Transformation within the MDI	Enterprise Interoperability	[198]
2015	A Model-based Approach for Resource Constrained Devices Energy Test and Simulation	Technological Innovation for Cloud-Based Engineering Systems	[213]
2016	Energy consumption awareness for resource-constrained devices	European Conference on Networks and Communications	[214]
2017	Multi-Criteria Analysis and Decision Methodology for the Selection of Internet-of-Things Hardware Platforms	Technological Innovation for Smart Systems	[203]
2017	A Multi-Criteria Decision Model for the Selection of a More Suitable Internet-of-Things Device	International Conference on Engineering, Technology and Innovation	[32]
	Other Publications:		
2014	Communication support for Petri nets based distributed controllers	IEEE 23rd International Symposium on Industrial Electronics	[218]
2014	A platform independent communication support for distributed controller systems modelled by Petri nets	12th IEEE International Conference on Industrial Informatics	[219]

9.2.2.2 Publications in Journals

The achievements regarding the publications in journal are presented in Table 9.3. Once more publications are organised by year (first column), and it is highlighted publication and journal name.

Table 9.3: Accomplished Publications in Journals.

Year:	Title:	Journal:	Ref:
2014	IoT Testbed Business Model	Advances in Internet of Things	[3]
2016	Energy Consumption Awareness for Resource-Constrained Devices: Extension to FPGA	Journal of Green Engineering	[195]
2019	Cyber-Physical Systems: A Multi-Criteria Assessment for Internet-of-Things (IoT) Systems	Enterprise Information Systems	[204]
2020	(IoT) Ecosystems Design: A Multi-Method, Multi-Criteria Assessment Methodology	Institute of Electrical and Electronics Engineers (IEEE) Internet of Things Journal	[215]

9.3 Future Work

Like all other PhD thesis, this work is not completely finalised. Besides some technical work needed to complete automate the proposed framework, new ideas had arisen during this research work that can be exploited even in new PhD thesis.

From a technical point of view, the author highlights two areas of the proposed framework, more specifically regarding IoT Systems formal specification. As mentioned, there is a wide diversity of IoT Systems, designed for different purposes, addressing a variety of application domains and services. Information is available from webpages, systems, modelling tools, datasheets, etc. Collect this data automatically needs more work, namely the use of ontologies to allow semantic identification and reasoning over important IoT System features, and consequently fill the framework models. The second aspect is related with software languages. This work addressed two software languages (C and nesC), proposing all the necessary changes to Meta-Models and presenting the respective enablers. However, there are a large number of software languages, which the proposed framework could embrace.

From a conceptual point of view, there is the concept of an energy profile of an IoT System. It is a wide theme, identified as one of the more important issues regarding Internet-of-Things (IoT). The author considers that the proposed formal specification of

an IoT System tackles a very important aspect in energy simulation, which is devices diversity. This formal description could be used as input for an energy analysis framework. Section 5.4.3 envisaged this issue, by proposing a high-level architecture of a possible approach to what the author identifies as a future, significant scientific contribution.

Furthermore, and already mentioned, it is foreseen the integration of this work with vf-OS, an European Project focus on provide a range of services for Factories of the Future integrating better manufacturing and logistics processes, to become a reference software managing factory' components system. It is envisaged, mainly, that Multi-Criteria Assessment of IoT Systems as a vf-OS Analytic Service, and IoT Systems formal specification as an useful vf-OS Driver.

Bibliography

[1] Merriam-Webster. *system*. Retrieved June 4, 2019. https://www.merriam-webster.com/dictionary/system. Merriam-webster, An Encyclopaedia Britannica, 2019.

[2] C. Dictionary. *system*. Retrieved June 4, 2019. https://dictionary.cambridge.org/dictionary/english/system. Cambridge Dictionary, 2019.

[3] E. M. Silva and P. Maló. "IoT Testbed Business Model." In: *Advances in Internet of Things* 4.4 (Oct. 2014), pp. 37–45. DOI: 10.4236/ait.2014.44006.

[4] I. Gartner. *Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020*. Access in March 16, 2015. http://www.gartner.com/newsroom/id/2636073. Gartner, Inc. Dec. 2013.

[5] A. Research. *More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020*. Access in July 2, 2015. https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/. Allied Business Intelligence, Inc, May 2013. (Visited on 02/07/2015).

[6] P. M. (UNINOVA), E. M. S. (UNINOVA), P. F. (UNINOVA), B. A. (UNINOVA), P. C. (EGM), A. G. U. of Surrey), S. Z. (CATR), G. M. (PERCEPTION), H. H. (CERT), N. D. (CSIR), and L. C. (CSIR). *Deliverable D3.1b Roadmaps for IoT Deployments*. FP7-288315 PROBE-IT "Pursuing ROadmaps and BEnchmarks for the Internet of Things". Project co-financed under the 7th framework program of the European Commission. Sept. 2013.

[7] D. Evans. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*. White Paper. Available from: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. CISCO Internet Business Solutions Group (IBSG), Apr. 2011.

[8] L. Atzori, A. Iera, and G. Morabito. "The Internet of Things: A Survey." In: *Computer Network* 54.15 (Oct. 2010), pp. 2787–2805. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2010.05.010.

[9] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac. "Internet of things: Vision, applications and research challenges." In: *Ad Hoc Networks* 10.7 (Sept. 2012), pp. 1497–1516. ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2012.02.016.

[10] M. Hempstead, M. J. Lyons, D. Brooks, and G.-Y. Wei. "Survey of Hardware Systems for Wireless Sensor Networks." In: *Journal of Low Power Electronics* 4.1 (Apr. 2008), pp. 11–20. ISSN: 1546-1998. DOI: 10.1166/jolpe.2008.156.

[11] J. Yick, B. Mukherjee, and D. Ghosal. "Wireless Sensor Network Survey." In: *Computer Networks: The International Journal of Computer and Telecommunications Networking* 52.12 (Aug. 2008), pp. 2292–2330. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2008.04.002.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Çayirci. "Wireless sensor networks: a survey." In: *Computer Networks* 38.4 (Mar. 2002), pp. 393–422. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(01)00302-4.

[13] D. for Communities and L. Government. *Multi-criteria analysis: a manual*. Manual. Available from: http://eprints.lse.ac.uk/12761/1/Multi-criteria_Analysis.pdf. Bressenden Place, London: Crown, Jan. 2009.

[14] Microchip. *New/Popular Microcontroller and Processors Products*. Retrieved June 11, 2019. https://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1005. Microchip Technology Inc., 2019.

[15] T. Instruments. *BOM & cross reference tool*. Retrieved June 11, 2019. https://bomcross.ti.com/en/. Texas Instruments Incorporated, 2019.

[16] S. K. Lee, M. Bae, and H. Kim. "Future of IoT Networks: A Survey." In: *Applied Sciences* 7.10 (2017). ISSN: 2076-3417. DOI: 10.3390/app7101072.

[17] A. Brogi and S. Forti. "QoS-Aware Deployment of IoT Applications Through the Fog." In: *IEEE Internet of Things Journal* 4.5 (Oct. 2017), pp. 1185–1192. ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2701408.

[18] F. Li, M. Vögler, M. Claeßens, and S. Dustdar. "Towards Automated IoT Application Deployment by a Cloud-Based Approach." In: *IEEE 6th International Conference on Service-Oriented Computing and Applications*. Dec. 2013, pp. 61–68. DOI: 10.1109/SOCA.2013.12.

[19] J. A. Lane and T. Bohn. "Using SysML modeling to understand and evolve systems of systems." In: *Systems Engineering* 16.1 (Mar. 2013), pp. 87–98. ISSN: 1098-1241. DOI: 10.1002/sys.21221.

[20] OMG. *SysML Open Source Project - What is SysML?* Retrieved June 7, 2019. https://sysml.org/. Object Management Group (OMG), June 2019.

[21] OGC. *Sensor Model Language (SensorML)*. Retrieved June 7, 2019. https://www.opengeospatial.org/standards/sensorml. Open Geospatial Consortium (OGC), Feb. 2014.

[22] W3C. *Semantic Sensor Network XG Final Report*. Retrieved June 26, 2019. https://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/. World Wide Web Consortium, June 2011.

[23] C. Agostinho. "Sustainability of systems interoperability in dynamic business networks." Available from: http://hdl.handle.net/10362/8582. Doctoral dissertation. Quinta da Torre, Caparica: Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Oct. 2012.

[24] K. E. Cambron and G. W. Evans. "Layout design using the analytic hierarchy process." In: *Computers and Industrial Engineering* 20.2 (Feb. 1991), pp. 211–229. ISSN: 0360-8352. DOI: 10.1016/0360-8352(91)90026-3.

[25] X. Wang and E. Triantaphyllou. "Ranking irregularities when evaluating alternatives by using some ELECTRE methods." In: *Omega* 36.1 (Feb. 2008), pp. 45–63. DOI: 10.1016/j.omega.2005.12.003.

[26] M. webster editors. *Scientific Method*. Access in November 11, 2016. https://www.merriam-webster.com/dictionary/scientificmethod. Merriam-webster, An Encyclopaedia Britannica.

[27] G. Dodig-Crnkovic. "Scientific Methods in Computer Science." In: *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*. Apr. 2002, pp. 126–130.

[28] S. D. Schafersman. *An introduction to science: Scientific thinking and the scientific method*. Accessed August 02, 2012. http://www.geo.sunysb.edu/esp/files/scientific-method.html. Department of Geology, Miami University. Jan. 1997.

[29] S. Buddies. *Steps of the Scientific Method*. Access in August 03, 2012. http://www.sciencebuddies.org/science-fair-projects/project_scientific_method.shtml. Science Buddies.

[30] O. Blakstad. *Research Methodology*. Access in January 09, 2017. https://explorable.com/research-methodology. Explorable.com. Mar. 2008.

[31] L. M. Camarinha-Matos. *Scientific Research Methodologies and Techniques — Unit 2: Scientific Method*. Access in August 02, 2012. http://www.uninova.pt/cam/teaching/SRMT/SRMTunit2.pdf. 2009.

[32] E. M. Silva, C. Agostinho, and R. Jardim-Goncalves. "A multi-criteria decision model for the selection of a more suitable Internet-of-Things device." In: *International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. June 2017, pp. 1268–1276. DOI: 10.1109/ICE.2017.8280026.

[33] J. A. Stankovic. "Research Directions for the Internet of Things." In: *Internet of Things Journal, IEEE* 1.1 (Feb. 2014), pp. 3–9. ISSN: 2327-4662. DOI: 10.1109/jiot.2014.2312291.

[34] M. B. (NEC), M. B. (ALBLF), N. B. (CFR), F. C. (UniS), C. J. (SIEMENS), J. D. L. (ALUBE), S. M. U. Carsten Magerkurth (SAP), A. N. F. IML), A. O. (CEA), M. T. (SAP), J. W. W. (SIEMENS), J. S. (CSD/SUni), and A. S. (UniWue). *Deliverable D1.5 — Final architectural reference model for the IoT v3.0*. FP7-257521 IoT-A "The Internet of Things – Architecture". Project co-financed under the 7th framework program of the European Commission. July 2013.

[35] E. Commission. *ICT30 — 2015 Internet of Things and Platforms for Connected Smart Objects*. Access in April 20, 2016. https://ec.europa.eu/digital-single-market/events/cf/ictpd14/item-display.cfm?id=12597. European Commission. 2014.

[36] E. Commission. *Horizon 2020 Work Programme 2016 -2017: 17. Cross-cutting activities (focus areas)*. Access in January 18, 2017. http://ec.europa.eu/research/participants/data/ref/h2020/wp/2016_2017/main/h2020-wp1617-focus_en.pdf. European Commission. 2016.

[37] E. Commission. *Horizon 2020 Work Programme 2018-2020: 5.i. Information and Communication Technologies*. Access in June 12, 2019. http://ec.europa.eu/research/participants/data/ref/h2020/wp/2018-2020/main/h2020-wp1820-leit-ict_en.pdf. European Commission. 2018.

[38] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. "A Dynamic Operating System for Sensor Nodes." In: *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*. MobiSys '05. New York, NY, USA: ACM, June 2005, pp. 163–176. ISBN: 1-931971-31-5. DOI: 10.1145/1067170.1067188.

[39] S. E. Díaz, J. C. Pérez, and J. F. Muñoz. "Survey of the State-of-the-Art in Flash-based Sensor Nodes." In: *Flash Memories*. Ed. by P. I. Stievano. Croatia: InTech, Sept. 2011. Chap. 6, pp. 113–136. ISBN: 978-953-307-272-2. DOI: 10.5772/19407.

[40] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. "System Architecture Directions for Networked Sensors." In: *SIGOPS Operating Systems Review* 34.5 (Dec. 2000), pp. 93–104. ISSN: 0163-5980. DOI: 10.1145/384264.379006.

[41] M. Hempstead, M. Welsh, and D. Brooks. "TinyBench: The Case For A Standardized Benchmark Suite for TinyOS Based Wireless Sensor Network Devices." In: *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*. LCN '04. Washington, DC, USA: IEEE Computer Society, Nov. 2004, pp. 585–586. ISBN: 0-7695-2260-2. DOI: 10.1109/LCN.2004.129.

[42] L. Nazhandali, M. Minuth, and T. Austin. "SenseBench: Toward an accurate evaluation of sensor network processors." In: *Proceedings of the IEEE International Workload Characterization Symposium*. IISWC-2005. IEEE, Oct. 2005, pp. 197–203. ISBN: 0780394615. DOI: 10.1109/IISWC.2005.1526017.

[43] M. O. Farooq and T. Kunz. "Operating Systems for Wireless Sensor Networks: A Survey." In: *Sensors* 11.6 (May 2011), pp. 5900–5930. ISSN: 1424-8220. DOI: 10.3390/s110605900.

[44] A. Corporation. *ATmega128 — 8-bit Atmel Microcontroller with 128KBytes In-System Programmable Flash*. Access in July 1, 2016. http://www.atmel.com/Images/doc2467.pdf. Atmel Corporation. June 2011.

[45] T. I. Incorporated. *MSP430^{TM} ultra-low-power Microcontrollers*. Access in July 1, 2016. http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/overview.page. Texas Instruments Incorporated. 2016.

[46] T. I. Incorporated. *CC2420 - Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee^{TM} Ready RF Transceiver*. Access in July 1, 2016. http://www.ti.com/product/cc2420. Texas Instruments Incorporated. 2016.

[47] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava. "Design Considerations for Solar Energy Harvesting Wireless Embedded Systems." In: *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*. IPSN '05. Piscataway, NJ, USA: IEEE Press, Apr. 2005. ISBN: 0-7803-9202-7.

[48] S. Roundy, P. K. Wright, and J. M. Rabaey. *Energy Scavenging for Wireless Sensor Networks: With Special Focus on Vibrations*. Boston, MA, USA: Springer Science & Business Media, 2004. ISBN: 978-1-4613-5100-9. DOI: 10.1007/978-1-4615-0485-6.

[49] M. Rahimi, H. Shah, G. Sukhatme, J. Heidemann, and D. Estrin. "Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 1. ICRA '03. Taipai, Taiwan: IEEE, Sept. 2003, pp. 19–24. DOI: 10.1109/ROBOT.2003.1241567.

[50] E. Kovacs, M. Bauer, J. Kim, J. Yun, F. Le Gall, and M. Zhao. "Standards-Based Worldwide Semantic Interoperability for IoT." In: *IEEE Communications Magazine* 54.12 (Dec. 2016), pp. 40–46. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.1600460CM.

[51] S. A. Weis. "RFID (radio frequency identification): Principles and applications." In: *System* 2.3 (2007), pp. 1–23.

[52] I. Square. *Near Field Communication Technology Standards*. Access in September 13, 2018. http://nearfieldcommunication.org/technology.html. Square, Inc. 2017.

[53] I. O. for Standardization (ISO) and I. E. C. (IEC). *ISO/IEC 18000-3:2010*. Tech. rep. 3. Available from: https://www.iso.org/standard/53424.html. ISO/IEC, Nov. 2010.

[54] Z. Alliance. *ZigBee Specification FAQ*. Access in September 13, 2018. https://web.archive.org/web/20130627172453/http://www.zigbee.org/Specifications/ZigBee/FAQ.aspx. Zigbee Alliance. 2013.

[55] M. T. Galeev. *Catching the Z-Wave*. Access in September 11, 2018. https://www.embedded.com/design/connectivity/4025721/Catching-the-Z-Wave. Embedded.com Aspencore Inc. 2006.

[56] I. Bluetooth SIG. *Radio Versions*. Access in March 19, 2019. https://www.bluetooth.com/bluetooth-technology/radio-versions. Bluetooth SIG, Inc. 2019.

[57] W.-F. Alliance. *Discover Wi-Fi*. Access in January 14, 2019. https://www.wi-fi.org/discover-wi-fi. Wi-Fi Alliance.

[58] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt. "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control." In: *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*. Apr. 2008, pp. 377–386. DOI: 10.1109/RTAS.2008.15.

[59] S. S.A. *Radio Technology Keypoints*. Access in January 14, 2019. https://www.sigfox.com/en/sigfox-iot-radio-technology. Sigfox S.A.

[60] L. Alliance. *What is LoRaWAN*. Access in January 14, 2019. https://lora-alliance.org/resource-hub/what-lorawantm. LoRa Alliance.

[61] R. Sanchez-Iborra, J. Sanchez-Gomez, J. Ballesta-Viñas, M.-D. Cano, and A. F. Skarmeta. "Performance Evaluation of LoRa Considering Scenario Conditions." In: *Sensors* 18.772 (Mar. 2018). DOI: 10.3390/s18030772.

[62] Y. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi. "A Primer on 3GPP Narrowband Internet of Things." In: *IEEE Communications Magazine* 55.3 (Mar. 2017), pp. 117–123. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600510CM.

[63] Sentineo. *NB-IOT or LORA, which technology to choose for your next IOT device*. Access in March 19, 2019. https://www.pietcallemeyn.be/sentineo/2018/04/30/nb-iot-or-lora-which-technology-to-choose-for-your-next-iot-device/. Sentineo. 2018.

[64] G. M. C. Centre. *Progress on 3GPP IoT*. Access in March 19, 2019. http://www.3gpp.org/news-events/3gpp-news/1766-iot_progress. 3GPP Mobile Competence Centre. 2016.

[65] U. A. Force. *GPS: The Global Positioning System*. Access in March 19, 2019. https://www.gps.gov/. U.S. Air Force. 2019.

[66] R. Meier and V. Cahill. "Taxonomy of distributed event-based programming systems." In: *The Computer Journal* 48.5 (Sept. 2005), pp. 585–586. ISSN: 0010-4620. DOI: 10.1093/comjnl/bxh120.

[67] C. Rathfelder, B. Klatt, K. Sachs, and S. Kounev. "Modeling event-based communication in component-based software architectures for performance predictions." In: *Software & Systems Modeling* 13.4 (2013), pp. 1291–1317. ISSN: 1619-1374. DOI: 10.1007/s10270-013-0316-x.

[68] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair. *Distributed Systems: Concepts and Design*. 5th. USA: Addison-Wesley Publishing Company, 2011. ISBN: 0132143011, 9780132143011.

[69] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Impact of mobility on Message Oriented Middleware (MOM) protocols for collaboration in transportation." In: *Proceedings of the IEEE 19th International Conference on Computer Supported Cooperative Work in Design*. CSCWD '15. IEEE, May 2015, pp. 115–120. DOI: 10.1109/CSCWD.2015.7230943.

[70] J. O'Hara. "Toward a Commodity Enterprise Middleware." In: *Queue* 5.4 (May 2007), pp. 48–55. ISSN: 1542-7730. DOI: 10.1145/1255421.1255424.

[71] A. Banks and R. Gupta. *MQTT Version 3.1.1*. Access in May 24, 2016. Retrieved from: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf. OASIS. Oct. 2014.

[72] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. Access in February 2, 2017. https://tools.ietf.org/html/rfc7252. Internet Engineering Task Force (IETF). 2014.

[73] *STOMP Protocol Specification, Version 1.2*. Access in April 11, 2016. Oct. 2012. URL: http://stomp.github.io/stomp-specification-1.2.html#Protocol_Overview.

[74] O. M. Group. *Data Distribution Service (DDS), version 1.4*. Tech. rep. formal/2015-04-10. Version 1.4. Available from: http://www.omg.org/spec/DDS/1.4/PDF. Needham, MA, USA: Object Management Group, Inc., Apr. 2015.

[75] G. Pardo-Castellote. "OMG Data-Distribution Service: Architectural Overview." In: *Proceedings of the 23rd International Conference on Distributed Computing Systems*. ICDCSW '03. Washington, DC, USA: IEEE Computer Society, May 2003, pp. 200–. ISBN: 0-7695-1921-0. URL: http://dl.acm.org/citation.cfm?id=839280.840571.

[76] T. A. S. Foundation. *Apache ActiveMQ*. Access in April 11, 2016. http://activemq.apache.org/. The Apache Software Foundation. 2011.

[77] A. Foster. *Messaging Technologies for the Industrial Internet and the Internet of Things Whitepaper: A Comparison Between DDS, AMQP, MQTT, JMS, REST, CoAP and XMPP*. Messaging Technologies Whitepaper. PrismTech, June 2015. URL: http://www.prismtech.com/sites/default/files/documents/Messaging-Whitepaper-040615_1.pdf.

[78] M. Iglesias-Urkia, A. Orive, and A. Urbieta. "Analysis of CoAP Implementations for Industrial Internet of Things: A Survey." In: *Procedia Computer Science* 109 (2017). 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal, pp. 188 – 195. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.05.323.

[79] N. Naik. "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP." In: *2017 IEEE International Systems Engineering Symposium (ISSE)*. Oct. 2017, pp. 1–7. DOI: 10.1109/SysEng.2017.8088251.

[80] M. Kovatsch. "CoAP for the Web of Things: From Tiny Resource-constrained Devices to the Web Browser." In: *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. UbiComp '13 Adjunct. Zurich, Switzerland: ACM, 2013, pp. 1495–1504. ISBN: 978-1-4503-2215-7. DOI: 10.1145/2494091.2497583.

[81] J. DizdareviMalć, F. Carpio, A. Jukan, and X. Masip-Bruin. "A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration." In: *ACM Comput. Surv.* 51.6 (Jan. 2019), 116:1–116:29. ISSN: 0360-0300. DOI: 10.1145/3292674.

[82] O. M. Group. *The Data Distribution Service, The Proven Data Connectivity Standard for the Internet of Things*. Access in April 11, 2016. Object Management Group, Inc. 2016. URL: http://portals.omg.org/dds/.

[83] P. S.-A. (editor). *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*. Access in March 29, 2016. Available from http://xmpp.org/rfcs/rfc3921.html. Jabber Software Foundation. Oct. 2004.

[84] N. Deakin. *Java Message Service (JMS) 2.0 ("Specification")*. Access in April 11, 2016. Oracle America, Inc. Mar. 2013. URL: http://download.oracle.com/otndocs/jcp/jms-2_0-fr-eval-spec/index.html.

[85] D. Ingram. "Reconfigurable Middleware for High Availability Sensor Systems." In: *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*. DEBS '09. Nashville, Tennessee: ACM, July 2009, 20:1–20:11. ISBN: 978-1-60558-665-6. DOI: 10.1145/1619258.1619285.

[86] A. Richardson. *RabbitMQ - An open source message broker that just work*. 2009. URL: http://www.rabbitmq.com/resources/RabbitMQcon.pdf.

[87] RabbitMQ. *Which protocols does RabbitMQ support?* Access in March 29, 2019. RabbitMQ. 2019. URL: https://www.rabbitmq.com/protocols.html.

[88] R. Bajpai, K. K. Dhara, and V. Krishnaswamy. "QPID: A Distributed Priority Queue with Item Locality." In: *Proceedings of the International Symposium on Parallel and Distributed Processing with Applications*. ISPA '08. Dec. 2008, pp. 215–223. DOI: 10.1109/ISPA.2008.90.

[89] J. Yin, S. He, F. Zhao, and S. Li. "Design and Implementation of Intelligent Load-Balancing Heterogeneous Data Source Middleware Based on ActiveMQ and XML." In: *Proceedings of the 2015 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration*. ICIICII '15. IEEE, Dec. 2015, pp. 255–258. DOI: 10.1109/ICIICII.2015.145.

[90] A. OLeary and B. Sherman. *An Introduction to Fusion Connect*. Conference Class, Las Vegas. 2017. URL: https://www.autodesk.com/autodesk-university/class/Introduction-Fusion-Connect-2017#downloads.

[91] S. Mathew. *Overview of Amazon Web Services*. AWS Whitepapers. Dec. 2018. URL: https://d1.awsstatic.com/whitepapers/aws-overview.pdf.

[92] G. E. Company. *Predix: The Industrial IoT Application Platform*. GE WhitePapers. 2018. URL: https://www.ge.com/digital/sites/default/files/download_assets/Predix-The-Industrial-Internet-Platform-Brief.pdf.

[93] Google. *Google Cloud IoT*. Access in March 26, 2019. https://cloud.google.com/solutions/iot/. Google. 2019.

[94] Google. *Cloud IoT Core*. Access in March 26, 2019. Google. 2019. URL: https://cloud.google.com/iot-core/.

[95] S. George. *Microsoft Azure IoT Suite —- Connecting Your Things to the Cloud*. Access in March 26, 2019. Microsoft. 2015. URL: https://azure.microsoft.com/en-gb/blog/microsoft-azure-iot-suite-connecting-your-things-to-the-cloud/.

[96] N. Berdy, R. Sherafat, R. Shahan, and D. Betts. *What is Azure IoT Hub?* Access in March 26, 2019. Microsoft. 2018. URL: https://docs.microsoft.com/en-us/azure/iot-hub/about-iot-hub.

[97] IBM. *IBM Watson IoT Platform*. Access in March 27, 2019. IBM. URL: https://www.ibm.com/us-en/marketplace/internet-of-things-cloud.

[98] IBM. *Watson IoT Platform*. Access in March 27, 2019. IBM. URL: https://www.ibm.com/cloud/watson-iot-platform.

[99] K. Companie. *Welcome to the Kaa IoT platform documentation!* Access in March 27, 2019. Kaa Companie. 2016. URL: https://kaaproject.github.io/kaa/docs/v0.10.0/Welcome/.

[100] K. Companie. *Everything an IoT platform should stand for*. Access in March 27, 2019. Kaa Companie. 2019. URL: https://www.kaaproject.org/.

[101] SiteWhere. *SiteWhere Platform*. Access in March 28, 2019. SiteWhere. 2019. URL: https://sitewhere.io/docs/2.0.0/platform/.

[102] SiteWhere. *Sending Data to SiteWhere*. Access in March 28, 2019. SiteWhere. 2019. URL: https://sitewhere.io/docs/2.0.0/guide/devices/sending-data.html.

[103] I. The MathWorks. *Learn More About ThingSpeak*. Access in March 28, 2019. The MathWorks, Inc. 2019. URL: https://thingspeak.com/pages/learn_more.

[104] I. The MathWorks. *The Open IoT Platform with MATLAB Analytics*. Access in March 28, 2019. The MathWorks, Inc. 2019. URL: https://www.mathworks.com/products/thingspeak.html.

[105] Ubidots. *IoT and Cloud tools to build your business*. Access in April 2, 2019. Ubidots. 2019. URL: https://ubidots.com/platform/.

[106] J. L. Hill. "System Architecture for Wireless Sensor Networks." Available from: http://eps2009.dj-inod.com/docs/09-02-01/system_architecture_for_wireless_sensor_networks.pdf. Doctoral dissertation. Berkeley, CA, EUA: University of California, Berkeley, Mar. 2003.

[107] A. Moschitta and I. Neri. "Power consumption Assessment in Wireless Sensor Networks." In: *ICT - Energy - Concepts Towards Zero - Power Information and Communication Technology*. Ed. by D. G. Fagas. InTech, 2014. Chap. 9. ISBN: 978-953-51-1218-1. DOI: 10.5772/57201.

[108] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. "TinyOS: An Operating System for Sensor Networks." In: *Ambient Intelligence*. Ed. by W. Weber, J. M. Rabaey, and E. Aarts. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 115–148. ISBN: 978-3-540-27139-0. DOI: 10.1007/3-540-27139-2_7.

[109] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. "The nesC Language: A Holistic Approach to Networked Embedded Systems." In: *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*. PLDI '03. San Diego, California, USA: ACM, June 2003, pp. 1–11. ISBN: 1-58113-662-5. DOI: 10.1145/781131.781133.

[110] K. Klues, C.-J. M. Liang, J. Paek, R. Musăloiu-E, P. Levis, A. Terzis, and R. Govindan. "TOSThreads: Thread-safe and Non-invasive Preemption in TinyOS." In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. SenSys '09. Berkeley, California: ACM, Nov. 2009, pp. 127–140. ISBN: 978-1-60558-519-2. DOI: 10.1145/1644038.1644052.

[111] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *RFC 4944 – Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. IETF RFC. Available from: https://tools.ietf.org/html/rfc4944. Sept. 2007.

[112] K. Lin and P. Levis. "Data Discovery and Dissemination with DIP." In: *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*. IPSN '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 433–444. ISBN: 978-0-7695-3157-1. DOI: 10.1109/IPSN.2008.17.

[113] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. "TinyDB: An Acquisitional Query Processing System for Sensor Networks." In: *ACM Transactions on Database Systems (TODS)* 30.1 (Mar. 2005), pp. 122–173. ISSN: 0362-5915. DOI: 10.1145/1061318.1061322.

[114] C. Karlof, N. Sastry, and D. Wagner. "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks." In: *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*. SenSys '04. Baltimore, MD, USA: ACM, Nov. 2004, pp. 162–175. ISBN: 1-58113-879-2. DOI: 10.1145/1031495.1031515.

[115] P. Levis, N. Lee, M. Welsh, and D. Culler. "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications." In: *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. SenSys '03. Los Angeles, California, USA: ACM, 2003, pp. 126–137. ISBN: 1-58113-707-9. DOI: 10.1145/958491.958506.

[116] A. Dunkels, B. Gronvall, and T. Voigt. "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors." In: *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*. LCN '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 455–462. ISBN: 0-7695-2260-2. DOI: 10.1109/LCN.2004.38.

[117] B. W. Kernighan. *The C Programming Language*. Ed. by D. M. Ritchie. 2nd. Prentice Hall Professional Technical Reference, 1988. ISBN: 0131103709.

[118] A. Dunkels and O. Schmidt. *Protothreads: Lightweight, Stackless Threads in C - SICS technical report, T2005:05*. Tech. rep. Available from: http://dunkels.com/adam/dunkels05protothreads.pdf. Swedish Institute of Computer Science, Mar. 2005.

[119] C. community. *Contiki: The Open Source OS for the Internet of Things*. Access in July 1, 2016. http://www.contiki-os.org/. Contiki community. 2016.

[120] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. *RPL: IPv6 Routing Protocol for Low Power and Lossy Networks,(RFC 6550)*. Standards Track. Available from: https://www.rfc-editor.org/rfc/rfc6550.txt. Internet Engineering Task Force (IETF), Mar. 2012.

[121] N. Tsiftes, J. Eriksson, and A. Dunkels. "Low-power Wireless IPv6 Routing with ContikiRPL." In: *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IPSN '10. Stockholm, Sweden: ACM, Apr. 2010, pp. 406–407. ISBN: 978-1-60558-988-6. DOI: 10.1145/1791212.1791277.

[122] N. Tsiftes, A. Dunkels, Z. He, and T. Voigt. "Enabling Large-scale Storage in Sensor Networks with the Coffee File System." In: *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. IPSN '09. Washington, DC, USA: IEEE Computer Society, Apr. 2009, pp. 349–360. ISBN: 978-1-4244-5108-1. URL: http://dl.acm.org/citation.cfm?id=1602165.1602197.

[123] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. "Cross-Level Sensor Network Simulation with COOJA." In: *Proceedings of the 31st IEEE Conference on Local Computer Networks*. SenseApp '06. Tampa, Florida, USA, Nov. 2006, pp. 641–648. DOI: 10.1109/LCN.2006.322172.

[124] L. Casado and P. Tsigas. "ContikiSec: A Secure Network Layer for Wireless Sensor Networks Under the Contiki Operating System." In: *Proceedings of the 14th Nordic Conference on Secure IT Systems: Identity and Privacy in the Internet Age*. NordSec '09. Oslo: Springer-Verlag, Oct. 2009, pp. 133–147. ISBN: 978-3-642-04765-7. DOI: 10.1007/978-3-642-04766-4_10.

[125] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. "MANTIS: System Support for multimodAl NeTworks of In-situ Sensors." In: *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*. WSNA '03. San Diego, CA, USA: ACM, 2003, pp. 50–59. ISBN: 1-58113-764-8. DOI: 10.1145/941350.941358.

[126] R. von Behren, J. Condit, and E. Brewer. "Why Events Are a Bad Idea (for High-concurrency Servers)." In: *Proceedings of the 9th Conference on Hot Topics in Operating Systems - Volume 9*. HOTOS'03. Lihue, Hawaii: USENIX Association, May 2003, pp. 4–. URL: http://dl.acm.org/citation.cfm?id=1251054.1251058.

[127] B. L. Titzer, D. K. Lee, and J. Palsberg. "Avrora: Scalable Sensor Network Simulation with Precise Timing." In: *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*. IPSN '05. Los Angeles, California: IEEE Press, Apr. 2005. ISBN: 0-7803-9202-7. URL: http://dl.acm.org/citation.cfm?id=1147685.1147768.

[128] R. Lajara, J. Pelegrí-Sebastiá, and J. J. P. Solano. "Power Consumption Analysis of Operating Systems for Wireless Sensor Networks." In: *Sensors* 10.6 (June 2010), pp. 5809–5826. ISSN: 1424-8220. DOI: 10.3390/s100605809.

[129] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh. "Simulating the Power Consumption of Large-scale Sensor Network Applications." In: *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*. SenSys '04. Baltimore, MD, USA: ACM, Nov. 2004, pp. 188–200. ISBN: 1-58113-879-2. DOI: 10.1145/1031495.1031518.

[130] E. Perla, A. O. Catháin, R. S. Carbajo, M. Huggard, and C. Mc Goldrick. "PowerTOSSIM Z: Realistic Energy Modelling for Wireless Sensor Network Environments." In: *Proceedings of the 3Nd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*. PM2HW2N '08. Vancouver, British Columbia, Canada: ACM, 2008, pp. 35–42. ISBN: 978-1-60558-239-9. DOI: 10.1145/1454630.1454636.

[131] M. S. E. Díaz. "A generic software architecture for portable applications in heterogeneous wireless sensor networks." Available from: http://hdl.handle.net/10016/9188. Doctoral dissertation. Calle Madrid: Universidad Carlos III de Madrid. Departamento de Informática, Mar. 2010.

[132] C. Organisation. *Mica 2*. Access in October 3, 2016. https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf. Crossbow Technology, Inc.

[133] T. Alliance. *TinyOS Home Page*. Access in October 3, 2016. http://www.tinyos.net/. TinyOS Alliance Steering Committee.

[134] C. Organisation. *Mica 2 Dot*. Access in October 3, 2016. https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2dot.pdf. Crossbow Technology, Inc.

[135] T. Bokareva. *Mini Hardware Survey*. Access in October 3, 2016. http://www.cse.unsw.edu.au/~sensar/hardware/hardware_survey.html.

[136] A. Rowe, F. Mokaya, M. Buevich, and P. Lazik. *Nano-RK*. Access in October 3, 2016. http://www.nano-rk.org/projects/nanork/wiki. Carnegie Mellon University. May 2011.

[137] A. Corporation. *Atmel AVR 8-bit and 32-bit Microcontrollers*. Access in October 3, 2016. http://www.atmel.com/products/microcontrollers/avr/default.aspx. Atmel Corporation.

[138] K. C. Park and D.-H. Shin. "Security assessment framework for IoT service." In: *Telecommunication Systems* 64.1 (Jan. 2017), pp. 193–209. ISSN: 1572-9451. DOI: 10.1007/s11235-016-0168-0.

[139] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan. "Internet of things (IoT) security: Current status, challenges and prospective measures." In: *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. Dec. 2015, pp. 336–341. DOI: 10.1109/ICITST.2015.7412116.

[140] Y. Chahid, M. Benabdellah, and A. Azizi. "Internet of things security." In: *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*. Apr. 2017, pp. 1–6. DOI: 10.1109/WITS.2017.7934655.

[141] X. Huang, P. Craig, H. Lin, and Z. Yan. "SecIoT: a security framework for the Internet of Things." In: *Security and Communication Networks* 9.16 (2016), pp. 3083–3094. DOI: 10.1002/sec.1259. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.1259.

[142] N. Wang, T. Jiang, W. Li, and S. Lv. "Physical-layer security in Internet of Things based on compressed sensing and frequency selection." In: *IET Communications* 11.9 (July 2017), pp. 1431–1437. DOI: 10.1049/iet-com.2016.1088.

[143] O. Vermesan and P. Friess. *Building the Hyperconnected Society: Internet of Things Research and Innovation Value Chains, Ecosystems and Markets*. Vol. 43. River Publishers Series in Communications. River Publishers, June 2015. ISBN: 9788793237995. DOI: 10.13052/rp-9788793237988.

[144] D. C. Schmidt. "Guest Editor's Introduction: Model-Driven Engineering." In: *IEEE Computer* 39.2 (Feb. 2006), pp. 25–31. ISSN: 0018-9162. DOI: 10.1109/MC.2006.58.

[145] A. F. Case. "Computer-aided Software Engineering (CASE): Technology for Improving Software Development Productivity." In: *SIGMIS Database* 17.1 (Sept. 1985), pp. 35–43. ISSN: 0095-0033. DOI: 10.1145/1040694.1040698.

[146] F.-I. Editors. *Computer-Aided Software Engineering*. Access in May 24, 2016. http://ithandbook.ffiec.gov/it-booklets/development-and-acquisition/development-procedures/software-development-techniques/computer-aided-software-engineering.aspx. Federal Financial Institution Examination Council (FFIEC).

[147] J. Hutchinson, M. Rouncefield, and J. Whittle. "Model-driven engineering practices in industry." In: *Proceeding of the 33rd international conference on Software engineering*. ICSE '11. Waikiki, Honolulu, HI-USA: ACM, 2011, pp. 633–642. ISBN: 978-1-4503-0445-0. DOI: 10.1145/1985793.1985882.

[148] J. Miller, J. Mukerji, M. Belaunde, F. Cummins, D. Dsouza, K. Duddy, W. E. Kaim, A. Kennedy, W. Frank, D. Frankel, R. Hauch, S. Hendryx, M. Hettinger, R. Hubert, D. Hybertson, S. Iyengar, J. Jourdan, T. Koch, A. Mallia, S. Mellor, J. Miller, J. Mischkinsky, C. Mullins, M. Oya, L. Rioux, P. Rivett, E. Seidewitz, B. Selic, J. Siegel, O. Sims, D. Smith, R. Soley, A. Tanaka, S. Tyndale-Biscoe, A. Watson, D. Weiseand, and B. Wood. *MDA Guide Version 1.0.1*. Tech. rep. omg/2003-06-01. Available from: http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf. Needham, MA, USA: Object Management Group, Inc., June 2003.

[149] C. Atkinson and T. Kühne. "Model-driven development: A metamodeling foundation." In: *IEEE Software* 20.5 (Sept. 2003), pp. 36–41. ISSN: 07407459. DOI: 10.1109/MS.2003.1231149.

[150] D. Ameller. "Considering Non-Functional Requirements in Model-Driven Engineering." Access in May 24, 2016. http://hdl.handle.net/2099.1/7192. Master's thesis. Universitat Politècnica de Catalunya Departament de Llenguatges i Sistemes Informàtics (LSI), June 2009.

[151] J. Bézivin. "On the Unification Power of Models." In: *Software and Systems Modeling (SoSyM)* 4.2 (May 2005), pp. 171–188. ISSN: 1619-1366. DOI: 10.1007/s10270-005-0079-0.

[152] T. Kühne. "What is a Model?" In: *Language Engineering for Model-Driven Software Development*. Ed. by J. Bezivin and R. Heckel. Dagstuhl Seminar Proceedings 04101. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2005, pp. 1–10. URL: http://drops.dagstuhl.de/opus/volltexte/2005/23.

[153] T. Kühne. "Matters of (meta-) modeling." In: *Software & Systems Modeling* 5.4 (July 2006), pp. 369–385. ISSN: 1619-1366. DOI: 10.1007/s10270-006-0017-9.

[154] M. webster editors. *Model*. Access in January 20, 2015. http://www.merriam-webster.com/dictionary/model. Merriam-webster, An Encyclopaedia Britannica.

[155] G. Doumeingts, A. Berre, J.-P. Bourey, and R. Grangel. *Deliverable DTG2.1: REPORT ON MODEL ESTABLISHMENT*. FP6 IST-508011 Interop NoE Project "Interoperability Research for Networked Enterprises Applications and Software Network of Excellence". Project co-financed under the 6th framework program of the European Commission. Dec. 2005.

[156] E. Seidewitz. "What models mean." In: *IEEE Software* 20.5 (Sept. 2003), pp. 26–32. ISSN: 07407459. DOI: 10.1109/MS.2003.1231147.

[157] O. M. Group. *OMG Unified Modeling LanguageTM (OMG UML), Superstructure*. Tech. rep. formal/2011-08-06. Version 2.4.1. Available from: http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/. Needham, MA, USA: Object Management Group, Inc., Aug. 2011.

[158] T. Clark, A. Evans, and S. Kent. "Engineering Modelling Languages: A Precise Meta-Modelling Approach." In: *Proceedings of the 5th International Conference on Fundamental Approaches to Software Engineering*. FASE '02. London, UK, UK: Springer-Verlag, Apr. 2002, pp. 159–173. ISBN: 3-540-43353-8. URL: http://dl.acm.org/citation.cfm?id=645370.651297.

[159] S. W. Ambler. *Agile Modeling (AM) Home Pag - Effective Practices for Modeling and Documentation*. Access in May 24, 2016. http://www.agilemodeling.com/. Ambysoft Inc. 2014.

[160] D. Thomas and B. M. Barry. "Model Driven Development: The Case for Domain Oriented Programming." In: *Proceeding in the Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*. OOPSLA '03. Anaheim, CA, USA: ACM, Oct. 2003, pp. 2–7. ISBN: 1-58113-751-6. DOI: 10.1145/949344.949346.

[161] J. Greenfield and K. Short. "Software factories: assembling applications with patterns, models, frameworks and tools." In: *Proceedings of the Companion of the 18th annual ACM SIGPLAN Conference on Object-oriented programming systems languages and applications*. Vol. 9. OOPSLA '03. Anaheim, CA, USA: ACM, Oct. 2003, pp. 16–27. ISBN: 1-58113-751-6. DOI: 10.1145/949344.949348.

[162] O. M. Group. *MDA - The Architecture of Choice for a Changing World*. Access in May 24, 2016. http://www.omg.org/mda/. Object Management Group, Inc.

[163] R. Soley and O. S. S. Group. *Model Driven Architecture*. Whitepaper. Available from: http://www.omg.org/mda/mda_files/model_driven_architecture.htm. Object Management Group, Nov. 2000.

[164] J. Bézivin. "From Object Composition to Model Transformation with the MDA." In: *Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS39)*. TOOLS '01. Washington, DC, USA: IEEE Computer Society, Aug. 2001, pp. 350–. URL: http://dl.acm.org/citation.cfm?id=882501.884684.

[165] J. Bézivin and O. Gerbé. "Towards a Precise Definition of the OMG/MDA Framework." In: *Proceedings of the 16th IEEE International Conference on Automated Software Engineering*. ASE '01. Washington, DC, USA: IEEE Computer Society, Nov. 2001, pp. 273–. DOI: 10.1109/ASE.2001.989813.

[166] A.-J. Berre, F. Liu, J. Xu, and B. Elvesaeter. "Model Driven Service Interoperability through Use of Semantic Annotations." In: *Proceedings of the International Conference on Interoperability for Enterprise Software and Applications*. IESA '09. China, Apr. 2009, pp. 90–96. DOI: 10.1109/I-ESA.2009.58.

[167] R. Grangel, M. Bigand, and J.-P. Bourey. "A UML profile as support for transformation of business process models at enterprise leve." In: *Proceedings of the First International Workshop on Model Driven Interoperability for Sustainable Information Systems*. MDISIS '08. Montpellier, France, June 2008, pp. 73–87. URL: http://ceur-ws.org/Vol-340/.

[168] Y. Singh and M. Sood. "Models and Transformations in MDA." In: *Proceeding of the First International Conference on Computational Intelligence, Communication Systems and Networks*. CICSYN '09. Indore, India: IEEE Computer Society, July 2009, pp. 253–258. ISBN: 978-0-7695-3743-6. DOI: 10.1109/CICSYN.2009.52.

[169] O. M. Group. *OMG Meta Object Facility (MOF) Core Specification*. Tech. rep. formal/2015-06-05. Version 2.5. Available from: http://www.omg.org/spec/MOF/2.5/PDF/. Needham, MA, USA: Object Management Group, Inc., June 2015.

[170] O. M. Group. *XML Metadata Interchange (XMI) Specification*. Tech. rep. formal/2015-06-07. Version 2.5.1. Available from: http://www.omg.org/spec/XMI/2.5.1/PDF/. Needham, MA, USA: Object Management Group, Inc., June 2015.

[171] S. Jörges and B. Steffen. "Exploiting Ecore's Reflexivity for Bootstrapping Domain-Specific Code-Generators." In: *35th Annual IEEE Software Engineering Workshop*. Oct. 2012, pp. 72–81. DOI: 10.1109/SEW.2012.14.

[172] F. Jouault and I. Kurtev. "On the Interoperability of Model-to-model Transformation Languages." In: *Science Computer Programming* 68.3 (Oct. 2007), pp. 114–137. ISSN: 0167-6423. DOI: 10.1016/j.scico.2007.05.005.

[173] O. M. Group. *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*. Tech. rep. formal/2016-06-03. Available from: http://www.omg.org/cgi-bin/doc?formal/2016-06-03.pdf. Needham, MA, USA: Object Management Group, Inc., June 2016.

[174] J. Bézivin. "Model Driven Engineering: An Emerging Technical Space." In: *Proceedings of the International Conference on Generative and Transformational Techniques in Software Engineering*. GTTSE'05. Braga, Portugal: Springer-Verlag, 2006, pp. 36–64. ISBN: 3-540-45778-X, 978-3-540-45778-7. DOI: 10.1007/11877028_2.

[175] R. N. Wabalickis. "Justification of FMS with the Analytic Hierarchy Process." In: *Journal of Manufacturing Systems* 7.3 (Jan. 1988), pp. 175 –182. ISSN: 0278-6125. DOI: 10.1016/0278-6125(88)90002-7.

[176] B. T. O. and M. E. L. "Multiattribute evaluation within a present framework and its relation to the analytic hierarchy process." In: *The Engineering Economist* 37.1 (1991), pp. 1–32. DOI: 10.1080/00137919108903055.

[177] M. Majumder. *Impact of Urbanization on Water Shortage in Face of Climatic Aberrations*. 1st ed. SpringerBriefs in Water Science and Technology 1. Springer Singapore, 2015. DOI: 10.1007/978-981-4560-73-3.

[178] M. Matos. *Ajuda Multicritério à Decisão - introdução*. Access in October 20, 2016. http://paginas.fe.up.pt/~mam/AD-intro.pdf. Faculdade de Engenharia da Universidade do Porto (FEUP). 2005.

[179] A. Kolios, V. Mytilinou, E. Lozano-Minguez, and K. Salonitis. "A Comparative Study of Multiple-Criteria Decision-Making Methods under Stochastic Inputs." In: *Energies* 9.7 (July 2016), p. 566. ISSN: 1996-1073. DOI: 10.3390/en9070566.

[180] D. Sabaei, J. Erkoyuncu, and R. Roy. "A Review of Multi-criteria Decision Making Methods for Enhanced Maintenance Delivery." In: *Procedia CIRP* 37 (Aug. 2015), pp. 30 –35. ISSN: 2212-8271. DOI: 10.1016/j.procir.2015.08.086.

[181] M. Aruldoss, T. M. Lakshmi, and V. P. Venkatesan. "A Survey on Multi Criteria Decision Making Methods and Its Applications." In: *American Journal of Information Systems* 1.1 (Dec. 2013), pp. 31–43. DOI: 10.12691/ajis-1-1-5.

[182] T. L. Saaty. "A scaling method for priorities in hierarchical structures." In: *Journal of Mathematical Psychology* 15.3 (June 1977), pp. 234 –281. ISSN: 0022-2496. DOI: 10.1016/0022-2496(77)90033-5.

[183] A. H. Aldlaigan and F. A. Buttle. "SYSTRA-SQ: a new measure of bank service quality." In: *International Journal of Service Industry Management* 13.4 (2002), pp. 362–381. DOI: 10.1108/09564230210445041.

[184] M. E. José Figueira Salvatore Greco. *Multiple Criteria Decision Analysis: State of the Art Surveys*. Ed. by S. Greco. International Series in Operations Research & Management Science. Vol. 78. 1. New York, USA: Springer-Verlag New York, 2005. ISBN: 978-0-387-23067-2. DOI: 10.1007/b100605.

[185] J. A. Alonso and M. T. Lamata. "Consistency in the Analytic Hierarchy Process: A New Approach." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 14.04 (Aug. 2006), pp. 445–459. DOI: 10.1142/S0218488506004114.

[186] H. Donegan and F. Dodd. "A note on saaty's random indexes." In: *Mathematical and Computer Modelling* 15.10 (Jan. 1991), pp. 135 –137. ISSN: 0895-7177. DOI: 10.1016/0895-7177(91)90098-R.

[187] J. P. Brans and P. Vincke. "Note-A Preference Ranking Organisation Method." In: *Management Science* 31.6 (June 1985), pp. 647–656. ISSN: 0025-1909. DOI: 10.1287/mnsc.31.6.647.

[188] K. Mela, T. Tiainen, and M. Heinisuo. "Comparative study of multiple criteria decision making methods for building design." In: *Advanced Engineering Informatics* 26.4 (Oct. 2012), pp. 716–726. ISSN: 1474-0346. DOI: 10.1016/j.aei.2012.03.001.

[189] J. R.S. C. Mateo. *Multi Criteria Analysis in the Renewable Energy Industry*. 1st ed. Green Energy and Technology 1. Springer-Verlag London, 2012. DOI: 10.1007/978-1-4471-2346-0.

[190] C. Macharis, J. Springael, K. D. Brucker, and A. Verbeke. "PROMETHEE and AHP: The design of operational synergies in multicriteria analysis.: Strengthening PROMETHEE with ideas of AHP." In: *European Journal of Operational Research* 153.2 (Mar. 2004). Management of the Future MCDA: Dynamic and Ethical Contributions, pp. 307 –317. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(03)00153-X.

[191] B. Roy. "The outranking approach and the foundations of electre methods." In: *Theory and Decision* 31.1 (July 1991), pp. 49–73. ISSN: 1573-7187. DOI: 10.1007/BF00134132.

[192] OMG. *OMG Systems Modeling Language*. Tech. rep. formal/2017-05-01. Version 1.5. Available from: https://sysml.org/docs/specs/OMGSysML-v1.5-17-05-01.pdf. Needham, MA, USA: Object Management Group, Inc., May 2017.

[193] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "Wireless sensor networks: a survey." In: *Computer Networks* 38.4 (2002), pp. 393 –422. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(01)00302-4.

[194] C. Consortium. *Customer in the Loop: Using Networked Devices enabled Intelligence for Proactive Customers Integration as Drivers of Integrated Enterprise*. Deliverable D1.3.2 Cuteloop Concept. FP7-2164220 Cuteloop, 2009.

[195] E. M. Silva, P. Maló, and M. Albano. "Energy Consumption Awareness for Resource-Constrained Devices: Extension to FPGA." In: *Journal of Green Engineering* 6.3 (July 2016), pp. 1 –27. ISSN: 1904-4720. DOI: 10.13052/jge1904-4720.631.

[196] KBSI. *IDEF0 Function Modeling Method*. Accessed March 22, 2019. Knowledge Based Systems, Inc., 2019. URL: http://www.idef.com/idefo-function_modeling_method/.

[197] C. Agostinho, J. Černý, and R. Jardim-Goncalves. "MDA-Based Interoperability Establishment Using Language Independent Information Models." In: *Enterprise Interoperability*. Ed. by M. van Sinderen, P. Johnson, X. Xu, and G. Doumeingts. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 146–160. ISBN: 978-3-642-33068-1.

[198] E. M. Silva, C. Agostinho, and R. Jardim-Gonçalves. "Achieving Interoperability via Model Transformation within the MDI." In: *Enterprise Interoperability*. John Wiley & Sons, Inc., Jan. 2013, pp. 171–180. ISBN: 9781118561942. DOI: 10.1002/9781118561942.ch26.

[199] J. Sarraipa, R. Jardim-Goncalves, and A. Steiger-Garcao. "MENTOR: an enabler for interoperable intelligent systems." In: *International Journal of General Systems* 39.5 (2010), pp. 557–573. DOI: 10.1080/03081079.2010.484278.

[200] L. F.C. S. Durão, M. M. Carvalho, S. Takey, P. A. Cauchick-Miguel, and E. Zancul. "Internet of Things process selection: AHP selection method." In: *The International Journal of Advanced Manufacturing Technology* 99.9 (Dec. 2018), pp. 2623–2634. ISSN: 1433-3015. DOI: 10.1007/s00170-018-2617-2.

[201] Y. P. Kondratenko, G. V. Kondratenko, and I. V. Sidenko. "Multi-Criteria Selection of the Wireless Communication Technology for Specialized IoT Network." In: *ICTERI Workshops*. 2018.

[202] Y. Kondratenko, G. Kondratenko, and I. Sidenko. "Multi-criteria decision making for selecting a rational IoT platform." In: *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. May 2018, pp. 147–152. DOI: 10.1109/DESSERT.2018.8409117.

[203] E. M. Silva and R. Jardim-Gonçalves. "Multi-Criteria Analysis and Decision Methodology for the Selection of Internet-of-Things Hardware Platforms." In: *Proceedings of the Technological Innovation for Smart Systems*. IFIP Advances in Information and Communication Technology. Accepted in January, 2017. Caparica, Portugal: Springer International Publishing, May 2017.

[204] E. M. Silva and R. Jardim-Goncalves. "Cyber-Physical Systems: a multi-criteria assessment for Internet-of-Things (IoT) systems." In: *Enterprise Information Systems* 0.0 (2019), pp. 1–20. DOI: 10.1080/17517575.2019.1698060.

[205] I. S. I. data. *ISO 10303-11 Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual*. Tech. rep. Available from: https://www.iso.org/standard/38047.html. International Organization for Standardization (ISO), Nov. 2004, p. 255.

[206] I. S. I. data. *ISO 10303-1 Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles*. Tech. rep. Available from: https://www.iso.org/standard/20579.html. International Organization for Standardization (ISO), Dec. 1994, p. 17.

[207] R. Jardim-Gonçalves, C. Agostinho, P. Maló, and A. Steiger-Garção. "Harmonising technologies in conceptual models representation." English. In: *International Journal of Product Lifecycle Management (IJPLM)* 2.2 (2007), pp. 187–205. ISSN: 1743-5110. DOI: 10.1504/IJPLM.2007.014279.

[208] H. Bruneliere and P. Guyard. *C 1.0*. Accessed July 25, 2019. AtlanMod, Aug. 2005. URL: http://web.emn.fr/x-info/atlanmod/index.php?title=Ecore#C_1.0.

[209] J. Ralo. "Representação da Linguagem nesC Usando Técnicas Baseadas em Modelos." Available in http://hdl.handle.net/10362/20680. Master Thesis. Caparica, Portugal: Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia, Jan. 2017.

[210] D. Gay, P. Levis, D. Culler, and E. Brewer. *nesC 1.3 Language Reference Manual*. Access in February 27, 2018. http://fossil.twicetwo.com/csci256.pl/doc/tip/doc/nesc1.3_ref.pdf. 2009.

[211] R. Pavlos. "Model Driven Development in Sensor Networks." Doctoral dissertation. Technical University of Crete, Electronic and Computer Engineering, June 2013.

[212] OGC. *SensorML 2.0 Examples*. Retrieved June 7, 2019. http://www.sensorml.com/sensorML-2.0/examples/. Open Geospatial Consortium (OGC), 2013.

[213] E. M. Silva, L. Gomes, J. Rodrigues, and P. Maló. "A Model-based Approach for Resource Constrained Devices Energy Test and Simulation." In: *Proceedings of the Technological Innovation for Cloud-Based Engineering Systems*. IFIP Advances in Information and Communication Technology. Caparica, Portugal: Springer International Publishing, Mar. 2015, pp. 345–354. ISBN: 978-3-319-16766-4. DOI: 10.1007/978-3-319-16766-4_37.

[214] E. M. Silva, P. Maló, and M. Albano. "Energy consumption awareness for resource-constrained devices." In: *Proceeding in the 25th Edition of the European Conference on Networks and Communications*. EuCNC 2016. Athens, Greece: IEEE, June 2016, pp. 74–78. ISBN: 978-1-5090-2893-1. DOI: 10.1109/EuCNC.2016.7561008.

[215] E. M. Silva and R. Jardim-Goncalves. "IoT Ecosystems Design: A Multi-Method, Multi-Criteria Assessment Methodology." In: *IEEE Internet of Things Journal* (2020), pp. 1–1. DOI: 10.1109/JIOT.2020.3011029.

[216] R. Poler, ICE, MASS, IKE, UNI, CMS, LYON2, ASC, ALM, APR, VS, CON, KBZ, and TARDY. *Virtual Factory Operating System Architecture (vf-OS): D2.2: Functional Specifications & Mockups*. Tech. rep. Available from: https://docs.wixstatic.com/ugd/0cf731_aea653224e80426f85f50210a10cdaf4.pdf. vf-OS, July 2017.

[217] O. Garcia and A. partners. *Virtual Factory Operating System Architecture (vf-OS): D2.1: Global Architecture Definition - Vs: 1.2.2*. Tech. rep. Available from: https://docs.wixstatic.com/ugd/0cf731_286b3f51e13141fa8aca27228b06aa87.pdf. vf-OS, June 2017.

[218] E. M. Silva, R. Campos-Rebelo, T. Hirashima, F. Moutinho, P. Maló, A. Costa, and L. Gomes. "Communication support for Petri nets based distributed controllers." In: *Proceeding in the 23rd International Symposium on Industrial Electronics*. ISIE '14. Istanbul, Turkey: IEEE, June 2014, pp. 1111–1116. DOI: 10.1109/ISIE.2014.6864769.

[219] E. M. Silva, P. Maló, and L. Gomes. "A platform independent communication support for distributed controller systems modelled by Petri nets." In: *Proceeding in the 12th IEEE International Conference on Industrial Informatics*. INDIN '2014. Porto Alegre, Brazil: IEEE, July 2014, pp. 88–93. DOI: 10.1109/INDIN.2014.6945489.

Appendix: IoTSAG Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="iotsag" nsURI=
    "http://IoTSystemsAssessment/IoTSystemsAnalysisGeneric"
    nsPrefix="IoTSAG">
  <eClassifiers xsi:type="ecore:EClass" name="PropertyDomain">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="domain" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Unit">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="unit" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="PropertyDomain" lowerBound="1"
        upperBound="-1" eType="#//PropertyDomain"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="unitType" lowerBound="1"
        eType="#//UnitTypes" defaultValueLiteral="STRING"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Property" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" ordered="false"
        upperBound="-1" eType="#//Annotation" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SingleProperty" eSuperTypes="#//Property">
    <eStructuralFeatures xsi:type="ecore:EReference" name="Unit" lowerBound="1" eType=
    "#//Unit"
        derived="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        iD="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="AggregationProperty" eSuperTypes="#//Property">
    <eStructuralFeatures xsi:type="ecore:EReference" name="composedBy" ordered="false"
        lowerBound="1" upperBound="-1" eType="#//Property" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="PropertyDomain" lowerBound="1"
        eType="#//PropertyDomain" derived="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Annotation">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description" ordered="false"
        lowerBound="1" eType="ecore:EDataType
        http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Definitions">
    <eStructuralFeatures xsi:type="ecore:EReference" name="propertyDomain" lowerBound="1"
        upperBound="-1" eType="#//PropertyDomain" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="unit" lowerBound="1" upperBound=
    "-1"
        eType="#//Unit" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" lowerBound="1"
        eType="#//Annotation" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SystemDescription" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="version" unique="false"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
```

```
        defaultValueLiteral="none"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="id" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        defaultValueLiteral="" iD="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" lowerBound="1"
        eType="#//Annotation" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="UnitTypes">
    <eLiterals name="STRING" literal="STRING"/>
    <eLiterals name="INTEGER" value="1"/>
    <eLiterals name="DOUBLE" value="2"/>
    <eLiterals name="FLOAT" value="3"/>
    <eLiterals name="CHARACTER" value="4"/>
    <eLiterals name="BOOLEAN" value="5"/>
    <eLiterals name="BYTE" value="6"/>
    <eLiterals name="LONG" value="7"/>
    <eLiterals name="SHORT" value="8"/>
  </eClassifiers>
</ecore:EPackage>
```

APPENDIX B

Appendix: RCSM Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="rcs" nsURI=
    "http://IoTSystemsAssessment/ResourceConstrainedSystem"
    nsPrefix="RCS">
  <eClassifiers xsi:type="ecore:EClass" name="ResourceConstrainedSystem" eSuperTypes=
  "../IoTSystemsAnalysisGeneric.ecore#//SystemDescription">
    <eAnnotations source="IoTSystemsAnalysisCoreModel.genmodel" references=
    "../IoTSystemsAnalysisGeneric.ecore#/"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="HardwareDescription" lowerBound="1"
        eType="ecore:EClass ResourceConstrainedSystemHardware.ecore#//HardwareModel"
        derived="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SoftwareDescription" unique="false"
        lowerBound="1" eType="#//SoftwareModel" derived="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="EnergyProfile" eType=
    "#//EnergyProfile"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="properties" upperBound="-1"
        eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Property" containment="true">
      <eAnnotations source="IoTSystemsAnalysisCoreModel.genmodel" references=
      "../IoTSystemsAnalysisGeneric.ecore#/"/>
    </eStructuralFeatures>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SoftwareModel" abstract="true" interface="true">
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" lowerBound="1"
        eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Annotation" containment=
        "true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="appVersion" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="properties" upperBound="-1"
        eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Property" containment="true"
        />
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="EnergyProfile" abstract="true" interface="true">
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" lowerBound="1"
        eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Annotation" containment=
        "true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="properties" upperBound="-1"
        eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Property" containment="true"
        />
  </eClassifiers>
</ecore:EPackage>
```

APPENDIX C

Appendix: RCSH Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="rcsh" nsURI=
    "http://IoTSystemsAssessment/ResourceConstrainedSystemHardware"
    nsPrefix="RCSH">
  <eClassifiers xsi:type="ecore:EClass" name="Device">
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" ordered="false"
        upperBound="-1" eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Annotation"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="isComposedBy" ordered="false"
        lowerBound="1" upperBound="-1" eType="#//Module" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="reference" ordered="false"
        lowerBound="1" eType="ecore:EDataType
        http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" unique=
    "false"
        lowerBound="1" eType="ecore:EDataType
        http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="version" ordered="false"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Component" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="contains" ordered="false"
        upperBound="-1" eType="#//Component" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="properties" ordered="false"
        lowerBound="1" upperBound="-1" eType="ecore:EClass
        ../IoTSystemsAnalysisGeneric.ecore#//Property"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" ordered="false"
        upperBound="-1" eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Annotation"
        containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Module">
    <eStructuralFeatures xsi:type="ecore:EReference" name="contains" ordered="false"
        lowerBound="1" upperBound="-1" eType="#//Component" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" ordered="false"
        upperBound="-1" eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Annotation"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="reference" ordered="false"
        lowerBound="1" eType="ecore:EDataType
        http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="version" ordered="false"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="HardwareModel">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" lowerBound=
    "1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" changeable=
        "false"
        defaultValueLiteral="Resource-Constrained System Hardware Model"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="describes" ordered="false"
        lowerBound="1" eType="#//Device" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="hwVersion" ordered="false"
        lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        defaultValueLiteral=""/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="modelVersion" ordered="false"
```

```
        lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        changeable="false" defaultValueLiteral="4.0-2019"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" lowerBound="1"
        eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Annotation" containment=
        "true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ProcessingUnit" eSuperTypes="#//Component"/>
  <eClassifiers xsi:type="ecore:EClass" name="Memory" eSuperTypes="#//Component"/>
  <eClassifiers xsi:type="ecore:EClass" name="Interfaces" abstract="true" eSuperTypes=
  "#//Component"/>
  <eClassifiers xsi:type="ecore:EClass" name="Security" eSuperTypes="#//Component"/>
  <eClassifiers xsi:type="ecore:EClass" name="PowerSupply" eSuperTypes="#//Component"/>
  <eClassifiers xsi:type="ecore:EClass" name="SensingUnit" eSuperTypes="#//Component"/>
  <eClassifiers xsi:type="ecore:EClass" name="Actuator" eSuperTypes="#//Component"/>
  <eClassifiers xsi:type="ecore:EClass" name="MachineMachine" eSuperTypes="#//Interfaces"/>
  <eClassifiers xsi:type="ecore:EClass" name="HumanMachine" eSuperTypes="#//Interfaces"/>
</ecore:EPackage>
```

APPENDIX D

Appendix: MCAM Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="mcam" nsURI=
    "http://IoTSystemsAssessment/MultiCriteriaAnalysisModel"
    nsPrefix="MCAM">
  <eClassifiers xsi:type="ecore:EClass" name="MultiCriteriaAnalysisModel">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="version" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        defaultValueLiteral="2.0-2019"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="objective" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="assessmentCriteria" lowerBound="1"
        upperBound="-1" eType="#//Criteria" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="DecisionMethod" lowerBound="1"
        upperBound="-1" eType="#//MultiCriteriaDecisionMethod"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="rankOutcome" upperBound="-1"
        eType="#//RankOutcome" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Criteria" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EReference" name="constraints" lowerBound="1"
        upperBound="-1" eType="#//Constraint" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="values" lowerBound="2"
        upperBound="-1" eType="#//CriteriaValue" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="Unit" lowerBound="1" eType=
    "ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Unit"
        derived="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CriteriaValue">
    <eStructuralFeatures xsi:type="ecore:EReference" name="ResourceConstrainedSystem"
        lowerBound="1" eType="ecore:EClass
        ../Device/ResourceConstrainedSystem.ecore#//ResourceConstrainedSystem"
        derived="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="DataValue" lowerBound="1"
        eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//SingleProperty" derived=
        "true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Constraint" abstract="true"/>
  <eClassifiers xsi:type="ecore:EClass" name="Optimization" eSuperTypes="#//Constraint">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType=
    "#//OptimizationType"
        defaultValueLiteral="MIN"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Availability" eSuperTypes="#//Constraint">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType=
    "#//AvailabilityType"
        defaultValueLiteral="MustHave"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Condition" abstract="true" eSuperTypes=
  "#//Constraint">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType=
    "#//ConditionType"
        defaultValueLiteral="LessThan"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="OptimizationType">
    <eLiterals name="MIN"/>
    <eLiterals name="MAX" value="1"/>
```

```
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="AvailabilityType">
    <eLiterals name="MustHave"/>
    <eLiterals name="CannotHave" value="1"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ConditionType">
    <eLiterals name="LessThan"/>
    <eLiterals name="Equal" value="1"/>
    <eLiterals name="GreaterThan" value="2"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Qualitative" eSuperTypes="#//Criteria"/>
  <eClassifiers xsi:type="ecore:EClass" name="Quantitative" eSuperTypes="#//Criteria"/>
  <eClassifiers xsi:type="ecore:EClass" name="QuantitativeCondition" eSuperTypes=
  "#//Condition">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="QualitativeCondition" eSuperTypes=
  "#//Condition">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" eType="ecore:EDataType
    http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="MultiCriteriaDecisionMethod" abstract="true"
      interface="true">
    <eStructuralFeatures xsi:type="ecore:EReference" name="annotation" upperBound="-1"
        eType="ecore:EClass ../IoTSystemsAnalysisGeneric.ecore#//Annotation" containment=
        "true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="RankOutcome">
    <eStructuralFeatures xsi:type="ecore:EReference" name="rank" lowerBound="2" upperBound=
    "-1"
        eType="#//Rank"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="DecisionMethod" lowerBound="1"
        eType="#//MultiCriteriaDecisionMethod"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Rank">
    <eStructuralFeatures xsi:type="ecore:EReference" name="ResourceConstrainedSystem"
        lowerBound="1" eType="ecore:EClass
        ../Device/ResourceConstrainedSystem.ecore#//ResourceConstrainedSystem"
        derived="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="rankingPosition" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EIntegerObject"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="outcomeValue" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EDoubleObject"/>
  </eClassifiers>
</ecore:EPackage>
```

APPENDIX E

Appendix: AHP Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="ahpm" nsURI=
    "http://IoTSystemsAssessment/AnalyticHierarchyProcessModel"
    nsPrefix="AHPM">
  <eAnnotations source="RandomConsistencyIndex" references="#//AHPModel">
    <details key="criteriaNumber_2" value="0"/>
    <details key="criteriaNumber_3" value="0.58"/>
    <details key="criteriaNumber_4" value="0.9"/>
    <details key="criteriaNumber_5" value="1.12"/>
    <details key="criteriaNumber_6" value="1.24"/>
    <details key="criteriaNumber_7" value="1.32"/>
    <details key="criteriaNumber_8" value="1.41"/>
    <details key="criteriaNumber_9" value="1.45"/>
    <details key="criteriaNumber_10" value="1.49"/>
  </eAnnotations>
  <eClassifiers xsi:type="ecore:EClass" name="AHPModel" eSuperTypes=
  "MultiCriteriaAnalysisModel.ecore#//MultiCriteriaDecisionMethod">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="method" unique="false"
        lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        changeable="false" defaultValueLiteral="Analytic Hierarchy Process"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="modelVersion" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        defaultValueLiteral="1.1-2019"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="criteriaComparison" lowerBound="1"
        eType="#//CriteriaComparison" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="consistencyRatioThreshold"
        lowerBound="1" eType="ecore:EDataType
        http://www.eclipse.org/emf/2002/Ecore#//EDoubleObject"
        defaultValueLiteral="0.10"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CriteriaComparison">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="scale" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        changeable="false" defaultValueLiteral="Saaty 1-9 scale"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="fromCriteria" lowerBound="2"
        upperBound="-1" eType="#//FromCriteria" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="FromCriteria">
    <eStructuralFeatures xsi:type="ecore:EReference" name="toCriteria" lowerBound="1"
        upperBound="-1" eType="#//ToCriteria" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="criteriaRef" lowerBound="1"
        eType="ecore:EClass MultiCriteriaAnalysisModel.ecore#//Criteria" derived="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ToCriteria">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="priorityValue" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EDoubleObject"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="criteriaRef" lowerBound="1"
        eType="ecore:EClass MultiCriteriaAnalysisModel.ecore#//Criteria" derived="true"/>
  </eClassifiers>
</ecore:EPackage>
```

APPENDIX F

Appendix: ELECTRE Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="electrem" nsURI=
    "http://IoTSystemsAssessment/ELECTREModel" nsPrefix="ELECTREM">
  <eClassifiers xsi:type="ecore:EClass" name="ELECTREModel" eSuperTypes=
  "MultiCriteriaAnalysisModel.ecore#//MultiCriteriaDecisionMethod">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="method" unique="false"
        lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        changeable="false" defaultValueLiteral="ELECTRE" unsettable="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="modelVersion" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        defaultValueLiteral="1.4-2019"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="weightVector" lowerBound="1"
        eType="#//CriteriaImportance" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="discordanceLevel" lowerBound="1"
        eType="#//Coalition" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="methodVersion" lowerBound="1"
        eType="#//ELECTRETypes" defaultValueLiteral="ELECTRE I"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="concordanceLevel" lowerBound="1"
        upperBound="5" eType="#//Coalition" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CriteriaImportance">
    <eStructuralFeatures xsi:type="ecore:EReference" name="electreCriterion" lowerBound="2"
        upperBound="-1" eType="#//ELECTRECriterion" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Threshold" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ELECTRECriterion">
    <eStructuralFeatures xsi:type="ecore:EReference" name="criteria" lowerBound="1"
        eType="ecore:EClass MultiCriteriaAnalysisModel.ecore#//Criteria" derived="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="weight" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EDoubleObject"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="preference" eType=
    "#//Discriminatory"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="indifference" eType=
    "#//Discriminatory"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="veto" eType="#//Veto" containment=
    "true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ELECTRETypes">
    <eLiterals name="ELECTRE_I" literal="ELECTRE I"/>
    <eLiterals name="ELECTRE_II" value="1" literal="ELECTRE II"/>
    <eLiterals name="ELECTRE_III" value="2" literal="ELECTRE III"/>
    <eLiterals name="ELECTRE_IV" value="3" literal="ELECTRE IV"/>
    <eLiterals name="ELECTRE_IS" value="4" literal="ELECTRE IS"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Coalition" eSuperTypes="#//Threshold"/>
  <eClassifiers xsi:type="ecore:EClass" name="Veto" eSuperTypes="#//Threshold"/>
  <eClassifiers xsi:type="ecore:EClass" name="Discriminatory" eSuperTypes="#//Threshold"/>
</ecore:EPackage>
```

APPENDIX G

Appendix: IoTSAC Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="iotsac" nsURI=
    "http://IoTSystemsAssessment/IoTSystemsAnalysisCore"
    nsPrefix="IoTSAC">
  <eClassifiers xsi:type="ecore:EClass" name="IoTSystemsAnalysis" eSuperTypes=
  "IoTSystemsAnalysisGeneric.ecore#//SystemDescription">
    <eStructuralFeatures xsi:type="ecore:EReference" name="MultiCriteriaDecision"
        lowerBound="1" upperBound="-1" eType="ecore:EClass
        MultiCriteriaAnalysisModel.ecore#//MultiCriteriaAnalysisModel"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="ResourceConstraintSystem"
        lowerBound="2" upperBound="-1" eType="ecore:EClass
        ResourceConstrainedSystem.ecore#//ResourceConstrainedSystem"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="TheSystemIoT" eSuperTypes=
  "IoTSystemsAnalysisGeneric.ecore#//SystemDescription">
    <eStructuralFeatures xsi:type="ecore:EReference" name="subSystem" lowerBound="1"
        upperBound="-1" eType="#//IoTSystemsAnalysis" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="definitions" lowerBound="1"
        upperBound="-1" eType="ecore:EClass IoTSystemsAnalysisGeneric.ecore#//Definitions"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="ResourceConstraintSystem"
        lowerBound="2" upperBound="-1" eType="ecore:EClass
        ResourceConstrainedSystem.ecore#//ResourceConstrainedSystem"/>
  </eClassifiers>
</ecore:EPackage>
```

APPENDIX H

Appendix: C Language Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="c" nsURI=
    "http://IoTSystemsAssessment/Ccode" nsPrefix="C">
  <eClassifiers xsi:type="ecore:EClass" name="ApplicationModel" eSuperTypes=
  "ResourceConstrainedSystem.ecore#//SoftwareModel">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="language" unique="false"
        lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        changeable="false" defaultValueLiteral="Language C"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="modelVersion" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString" changeable=
        "false"
        defaultValueLiteral="1.1-2019"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="functions" lowerBound="1"
        upperBound="-1" eType="#//CFunction" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CStructureContents">
    <eStructuralFeatures xsi:type="ecore:EReference" name="sc_container" ordered="false"
        eType="#//CStructured" eOpposite="#//CStructured/contains"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CClassifier"/>
  <eClassifiers xsi:type="ecore:EClass" name="CDataType" eSuperTypes="#//CClassifier"/>
  <eClassifiers xsi:type="ecore:EClass" name="CIntegral" eSuperTypes="#//CDataType"/>
  <eClassifiers xsi:type="ecore:EClass" name="CFloating" eSuperTypes="#//CDataType"/>
  <eClassifiers xsi:type="ecore:EClass" name="CBitField" eSuperTypes="#//CDataType"/>
  <eClassifiers xsi:type="ecore:EClass" name="CVoid" eSuperTypes="#//CDataType"/>
  <eClassifiers xsi:type="ecore:EClass" name="CEnumeration" eSuperTypes="#//CIntegral"/>
  <eClassifiers xsi:type="ecore:EClass" name="CInt" eSuperTypes="#//CIntegral"/>
  <eClassifiers xsi:type="ecore:EClass" name="CChar" eSuperTypes="#//CIntegral"/>
  <eClassifiers xsi:type="ecore:EClass" name="CDouble" eSuperTypes="#//CFloating"/>
  <eClassifiers xsi:type="ecore:EClass" name="CFloat" eSuperTypes="#//CFloating"/>
  <eClassifiers xsi:type="ecore:EClass" name="CLongDouble" eSuperTypes="#//CFloating"/>
  <eClassifiers xsi:type="ecore:EClass" name="CUnsignedInt" eSuperTypes="#//CInt"/>
  <eClassifiers xsi:type="ecore:EClass" name="CLong" eSuperTypes="#//CInt"/>
  <eClassifiers xsi:type="ecore:EClass" name="CLongLong" eSuperTypes="#//CInt"/>
  <eClassifiers xsi:type="ecore:EClass" name="CShort" eSuperTypes="#//CInt"/>
  <eClassifiers xsi:type="ecore:EClass" name="CSignedChar" eSuperTypes="#//CChar"/>
  <eClassifiers xsi:type="ecore:EClass" name="CUnsignedChar" eSuperTypes="#//CChar"/>
  <eClassifiers xsi:type="ecore:EClass" name="CWChar" eSuperTypes="#//CChar"/>
  <eClassifiers xsi:type="ecore:EClass" name="UnsignedLong" eSuperTypes="#//CUnsignedInt"/>
  <eClassifiers xsi:type="ecore:EClass" name="UnsignedLongLong" eSuperTypes="#//CUnsignedInt"
  />
  <eClassifiers xsi:type="ecore:EClass" name="UnsignedShort" eSuperTypes="#//CUnsignedInt"/>
  <eClassifiers xsi:type="ecore:EClass" name="Derived" eSuperTypes="#//CClassifier"/>
  <eClassifiers xsi:type="ecore:EClass" name="CStructured" eSuperTypes="#//CClassifier
  #//CStructureContents">
    <eStructuralFeatures xsi:type="ecore:EReference" name="contains" ordered="false"
        upperBound="-1" eType="#//CStructureContents" containment="true" eOpposite=
        "#//CStructureContents/sc_container"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CStruct" eSuperTypes="#//CStructured"/>
  <eClassifiers xsi:type="ecore:EClass" name="CUnion" eSuperTypes="#//CStructured"/>
  <eClassifiers xsi:type="ecore:EClass" name="CSourceText">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="source" ordered="false"
        unique="false" lowerBound="1" eType="#//PrimitiveTypes/String"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="fileName" ordered="false"
        unique="false" lowerBound="1" eType="#//PrimitiveTypes/String"/>
  </eClassifiers>
```

```
  <eClassifiers xsi:type="ecore:EClass" name="CTypedElement">
    <eStructuralFeatures xsi:type="ecore:EReference" name="type" ordered="false" lowerBound=
    "1"
        eType="#//CClassifier"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="source" ordered="false"
        lowerBound="1" eType="#//CSourceText"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CStructuralFeature" eSuperTypes=
  "#//CStructureContents #//CTypedElement"/>
  <eClassifiers xsi:type="ecore:EClass" name="CField" eSuperTypes="#//CStructuralFeature"/>
  <eClassifiers xsi:type="ecore:EClass" name="CParameter" eSuperTypes="#//CTypedElement">
    <eStructuralFeatures xsi:type="ecore:EReference" name="behavioralFeature" ordered="false"
        eType="#//BehavioralFeature" eOpposite="#//BehavioralFeature/parameters"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="BehavioralFeature">
    <eStructuralFeatures xsi:type="ecore:EReference" name="parameters" ordered="false"
        upperBound="-1" eType="#//CParameter" containment="true" eOpposite=
        "#//CParameter/behavioralFeature"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CFunction" eSuperTypes="#//BehavioralFeature">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="isVarArg" ordered="false"
        unique="false" lowerBound="1" eType="#//PrimitiveTypes/Boolean"/>
  </eClassifiers>
  <eSubpackages name="PrimitiveTypes" nsURI="http://IoTSystemsAssessment/PrimitiveTypes"
      nsPrefix="primitiveTypes">
    <eClassifiers xsi:type="ecore:EDataType" name="Integer" instanceTypeName="integer"/>
    <eClassifiers xsi:type="ecore:EDataType" name="String" instanceTypeName="string"/>
    <eClassifiers xsi:type="ecore:EDataType" name="Boolean" instanceClassName="boolean"/>
    <eClassifiers xsi:type="ecore:EDataType" name="Double" instanceClassName="double"/>
  </eSubpackages>
</ecore:EPackage>
```

APPENDIX I

Appendix: nesC Language Ecore Representation

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="nesc" nsURI=
    "http://IoTSystemsAssessment/NetworkEmbeddedSystemC"
    nsPrefix="nesC">
  <eClassifiers xsi:type="ecore:EClass" name="Statement" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EReference" name="body" eType="#//Body" containment=
    "true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ElementType">
    <eLiterals name="Generic" value="1"/>
    <eLiterals name="Normal"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="OperatorType">
    <eLiterals name="none"/>
    <eLiterals name="And" value="1"/>
    <eLiterals name="Or" value="2"/>
    <eLiterals name="Not" value="3"/>
    <eLiterals name="BitAnd" value="4"/>
    <eLiterals name="BitOr" value="5"/>
    <eLiterals name="BitXor" value="6"/>
    <eLiterals name="BitNot" value="7"/>
    <eLiterals name="Greater" value="8"/>
    <eLiterals name="GreaterEqual" value="9"/>
    <eLiterals name="Lesser" value="10"/>
    <eLiterals name="LesserEqual" value="11"/>
    <eLiterals name="LeftShift" value="12"/>
    <eLiterals name="RightShift" value="13"/>
    <eLiterals name="Plus" value="14"/>
    <eLiterals name="Minus" value="15"/>
    <eLiterals name="Multiply" value="16"/>
    <eLiterals name="Divide" value="17"/>
    <eLiterals name="Modulus" value="18"/>
    <eLiterals name="Assign" value="19"/>
    <eLiterals name="Equal" value="20"/>
    <eLiterals name="returnValue" value="21"/>
    <eLiterals name="void" value="22"/>
    <eLiterals name="MethodCall" value="23"/>
    <eLiterals name="Init" value="24"/>
    <eLiterals name="EnumerationAcc" value="25"/>
    <eLiterals name="ClassAcc" value="26"/>
    <eLiterals name="PathAcc" value="27"/>
    <eLiterals name="EmitEvent" value="28"/>
    <eLiterals name="NewFrame" value="29"/>
    <eLiterals name="DeleteFrame" value="30"/>
    <eLiterals name="EventAcc" value="31"/>
    <eLiterals name="NotEqual" value="32"/>
    <eLiterals name="Cast" value="33"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="InternalType">
    <eLiterals name="none"/>
    <eLiterals name="int" value="1"/>
    <eLiterals name="int8_t" value="2"/>
    <eLiterals name="int32_t" value="3"/>
    <eLiterals name="uint8_t" value="4"/>
    <eLiterals name="uint16_t" value="5"/>
    <eLiterals name="uint32_t" value="6"/>
    <eLiterals name="bool" value="7"/>
```

```
  <eLiterals name="char" value="8"/>
  <eLiterals name="float" value="9"/>
  <eLiterals name="double" value="10"/>
  <eLiterals name="void" value="11"/>
  <eLiterals name="pointer" value="12"/>
  <eLiterals name="struct" value="13"/>
  <eLiterals name="enumDeclaration" value="14"/>
  <eLiterals name="reference" value="15"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="If" eSuperTypes="#//Statement">
  <eStructuralFeatures xsi:type="ecore:EReference" name="condition" lowerBound="1"
      eType="#//Expression" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="elseBody" eType="#//Body"
      containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Atomic" eSuperTypes="#//Statement">
  <eStructuralFeatures xsi:type="ecore:EReference" name="statement" eType="#//Statement"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="For" eSuperTypes="#//Statement">
  <eStructuralFeatures xsi:type="ecore:EReference" name="condition" lowerBound="1"
      eType="#//Expression" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="int" lowerBound="1" eType=
  "#//Expression"
      containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="step" lowerBound="1" eType=
  "#//Expression"
      containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="While" eSuperTypes="#//Statement">
  <eStructuralFeatures xsi:type="ecore:EReference" name="condition" lowerBound="1"
      eType="#//Expression" containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Component">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType=
  "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType=
  "#//ElementType"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="provides" upperBound="-1"
      eType="#//Interface" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="uses" upperBound="-1" eType=
  "#//Interface"
      containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasDeclarations" upperBound="-1"
      eType="#//Variable" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="identifier" eType=
  "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasHeaders" upperBound="-1"
      eType="#//Header" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasParameters" upperBound="-1"
      eType="#//Argument"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasBareCommands" upperBound="-1"
      eType="#//Command" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasBareEvent" upperBound="-1"
      eType="#//Event" containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="EndPoint">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="identifier" lowerBound="1"
      eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
</eClassifiers>
```

```
<eClassifiers xsi:type="ecore:EClass" name="Wiring">
  <eStructuralFeatures xsi:type="ecore:EReference" name="provider" lowerBound="1"
      eType="#//EndPoint" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="user" lowerBound="1" eType=
  "#//EndPoint"
      containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Configuration" eSuperTypes="#//Component">
  <eStructuralFeatures xsi:type="ecore:EReference" name="wires" upperBound="-1"
      eType="#//Wiring" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="implementation" upperBound="-1"
      eType="#//Component" containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Module" eSuperTypes="#//Component">
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasVariables" upperBound="-1"
      eType="#//Variable" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="implements" upperBound="-1"
      eType="#//Method"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="moduleID" eType="ecore:EDataType
  http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Method" abstract="true">
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasBody" eType="#//Body"
      containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasParameters" upperBound="-1"
      eType="#//Variable" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType=
  "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="returnType" lowerBound="1"
      eType="#//VariableTypes" containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Interface">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType=
  "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType=
  "#//ElementType"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="identifier" eType=
  "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasEvents" upperBound="-1"
      eType="#//Event" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasCommands" upperBound="-1"
      eType="#//Command" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasParameters" upperBound="-1"
      eType="#//Argument"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Expression" eSuperTypes="#//Statement
#//Argument">
  <eStructuralFeatures xsi:type="ecore:EReference" name="left" eType="#//Argument"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="right" eType="#//Argument"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasArguments" upperBound="2"
      eType="#//Argument" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="operator" lowerBound="1"
      eType="#//OperatorType"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Argument" abstract="true"/>
<eClassifiers xsi:type="ecore:EClass" name="Literal" eSuperTypes="#//Argument">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="literal" lowerBound="1"
      eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
</eClassifiers>
```

```
<eClassifiers xsi:type="ecore:EClass" name="MethodRef" eSuperTypes="#//Argument">
  <eStructuralFeatures xsi:type="ecore:EReference" name="method" lowerBound="1"
      eType="#//Method"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasParameters" upperBound="-1"
      eType="#//Variable"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Variable" eSuperTypes="#//Argument">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType=
  "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="typedef" lowerBound="1"
      eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="InitExpression" eType=
  "#//Expression"
      containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="type" lowerBound="1" eType=
  "#//VariableTypes"
      containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Array" eSuperTypes="#//Variable">
  <eStructuralFeatures xsi:type="ecore:EReference" name="dimension" lowerBound="1"
      eType="#//Argument"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Struct" eSuperTypes="#//Variable">
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasStructs" upperBound="-1"
      eType="#//Struct"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasVariables" upperBound="-1"
      eType="#//Variable" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="struct_name" eType=
  "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Enums" eSuperTypes="#//Variable">
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasVariables" upperBound="-1"
      eType="#//Variable" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="enum_name" eType="ecore:EDataType
  http://www.eclipse.org/emf/2002/Ecore#//EString"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Header">
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasVariables" upperBound="-1"
      eType="#//Variable" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="includes" upperBound="-1"
      eType="#//Header" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasFunctions" upperBound="-1"
      eType="#//InternalFunction" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType
  http://www.eclipse.org/emf/2002/Ecore#//EString"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Body">
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasVariables" upperBound="-1"
      eType="#//Variable" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="hasStatements" upperBound="-1"
      eType="#//Statement" containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Task" eSuperTypes="#//Method">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="post" lowerBound="1" eType=
  "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="InternalFunction" eSuperTypes="#//Method"/>
<eClassifiers xsi:type="ecore:EClass" name="Command" eSuperTypes="#//Method">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="async" lowerBound="1" eType=
```

```
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Event" eSuperTypes="#//Method">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="async" lowerBound="1" eType=
    "ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="BinaryModule" eSuperTypes="#//Component"/>
  <eClassifiers xsi:type="ecore:EClass" name="VariableTypes" abstract="true"/>
  <eClassifiers xsi:type="ecore:EClass" name="ExternalTypes" eSuperTypes="#//VariableTypes">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType=
    "#//ExternalType"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="InternalTypes" eSuperTypes="#//VariableTypes">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" lowerBound="1" eType=
    "#//InternalType"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ExternalType">
    <eLiterals name="none"/>
    <eLiterals name="nx_int8_t" value="1"/>
    <eLiterals name="nx_uint8_t" value="2"/>
    <eLiterals name="nxle_int8_t" value="3"/>
    <eLiterals name="nxle_uint8_t" value="4"/>
    <eLiterals name="nx_int16_t" value="5"/>
    <eLiterals name="nx_uint16_t" value="6"/>
    <eLiterals name="nxle_int16_t" value="7"/>
    <eLiterals name="nxle_uint16_t" value="8" literal="nxle_uint16_t"/>
    <eLiterals name="nx_int32_t" value="9" literal="nx_int32_t"/>
    <eLiterals name="nx_uint32_t" value="10"/>
    <eLiterals name="nxle_int32_t" value="11"/>
    <eLiterals name="nxle_uint32_t" value="12"/>
    <eLiterals name="nx_int64_t" value="13"/>
    <eLiterals name="nx_uint64_t" value="14"/>
    <eLiterals name="nxle_int64_t" value="15"/>
    <eLiterals name="nxle_uint64_t" value="16"/>
    <eLiterals name="nx_struct" value="17"/>
    <eLiterals name="nx_union" value="18"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ApplicationModel" eSuperTypes=
  "ResourceConstrainedSystem.ecore#//SoftwareModel">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="modelVersion" ordered="false"
        lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        changeable="false" defaultValueLiteral="2.0-2019"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="configuration" ordered="false"
        eType="#//Component" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="language" unique="false"
        lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        changeable="false" defaultValueLiteral="nesC"/>
  </eClassifiers>
</ecore:EPackage>
```

APPENDIX J

Appendix: XML File of a SensorML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<sml:PhysicalComponent gml:id="MY_SENSOR"
   xmlns:sml="http://www.opengis.net/sensorml/2.0"
   xmlns:swe="http://www.opengis.net/swe/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xsi:schemaLocation="http://www.opengis.net/sensorml/2.0
   http://schemas.opengis.net/sensorml/2.0/sensorML.xsd">
   <!-- ================================================ -->
   <!--                  System Description              -->
   <!-- ================================================ -->
   <gml:description>simple thermometer with time tag</gml:description>
   <gml:identifier codeSpace="uniqueID">urn:meteofrance:stations:76455</gml:identifier>

   <!-- metadata deleted for brevity sake -->

   <!-- ================================================ -->
   <!--             Observed Property = Output           -->
   <!-- ================================================ -->
   <sml:outputs>
      <sml:OutputList>
         <sml:output name="temp">
            <sml:DataInterface>
               <sml:data>
                  <swe:DataStream>
                      <!-- describe output -->
                     <swe:elementType name="temperature">
                         <swe:Quantity
                                definition=
                                "http://mmisw.org/ont/cf/parameter/air_temperature">
                            <swe:uom code="Cel"/>
                         </swe:Quantity>
                     </swe:elementType>
                     <!-- simple text encoding  -->
                     <swe:encoding>
                        <swe:TextEncoding tokenSeparator="," blockSeparator=" "/>
                     </swe:encoding>
                     <!-- reference the values at a RESTful resource -->
                     <!-- returns latest measurement or continues to send new values through
                     open http pipe -->
                     <swe:values xlink:href="http://myServer.com:4563/sensor/02080"/>

                  </swe:DataStream>
               </sml:data>
            </sml:DataInterface>
         </sml:output>
      </sml:OutputList>
   </sml:outputs>
   <!-- ================================================ -->
   <!--                  Station Location                -->
   <!-- ================================================ -->
   <sml:position>
      <gml:Point gml:id="stationLocation" srsName=
      "http://www.opengis.net/def/crs/EPSG/0/4326">
         <gml:coordinates>47.8 88.56</gml:coordinates>
      </gml:Point>
   </sml:position>
</sml:PhysicalComponent>
```